# YARF: An Open-Ended Framework for Robot Road Following

Karl Kluge

24 February 1993
CMU-CS-93-104



School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

*Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy*

**Thesis Committee:**
Chuck Thorpe, Chair
Takeo Kanade
Tom Mitchell
Ed Riseman, University of Massachusetts, Amherst





This research was partially supported by contracts from the Defense Advanced Research Projects Agency, entitled "Robot System Development and Testing" (monitored by TACOM), and "Perception for Outdoor Navigation" (monitored by TEC).

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of DARPA or the U.S. Government.

**ACM Computing Reviews Keywords:** 1.2.10 Vision and Scene Understanding, 1.2.9 Robotics, 1.4.6 Segmentation, 1.4.8 Scene Analysis, 1.4.10 Image Representation

Carnegie Mellon

School of Computer Science

# DOCTORAL THESIS
in the field of
Computer Science

## *YARF: An Open-Ended Framework for Robot Road Following*

**KARL KLUGE**



Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy



**ACCEPTED:**

Charles E Thorpe — THESIS COMMITTEE CHAIR — 2/25/93 — DATE

J Mens — DEPARTMENT HEAD — 3/9/93 — DATE

**APPROVED:**

R. Rashid — DEAN — 3/11/93 — DATE

# YARF: An Open-Ended Framework for Robot Road Following

## CMU-CS-93-104

**Karl Kluge**

February 1993 - Thesis

This thesis describes YARF *(Yet Another Road Follower)*, a vision-based system for autonomous road following. Video data from a camera mounted on a robot vehicle is fed into a computer, which analyzes the image data to locate the position of the vehicle relative to the road. The computer then issues commands to actuators attached to the throttle, brakes, and steering in order to drive the vehicle along the road. YARF has been extensively tested using a combination of open- and closed-loop runs on testbed vehicles, simulation, and data from videotapes. YARF provides a set of perception capabilities to locate the position of the vehicle relative to the road, to detect changes in the lane structure of the road, to navigate through intersections given a model of the intersection geometry, and to extract the lane structure of the road without a prior model.

The central theme of YARF is that using richer models improves road following performance. Models of geometric structure, of road appearance, and of segmentation performance all simplify processing and contribute to improve reliability. YARF uses road models in several ways:

- Model drive segmentation.
- Exploitation of model coherence to avoid the influence of contaminating data.
- Data driven recognition of model changes.

While YARF assumes that the models of road structure used are generated off line, the thesis also presents an algorithm designed to automatically extract much of the needed model information. The algorithm uses a weak domain model to filter a noisy image segmentation, extracting both feature geometry and type.

**Keywords:** VISION AND SCENE UNDERSTANDING, ROBOTICS, SEGMENTATION, SCENE ANALYSIS, IMAGE REPRESENTATION

**(95 pages)**

# TECHNICAL REPORTS 1993 - NEW ADDITIONS

## School of Computer Science

**CMU-CS-93-104** **YARF: AN OPEN-ENDED FRAMEWORK FOR ROBOT ROAD FOLLOWING**
Karl Kluge
*February 1993*
KEYWORDS: Vision and scene understanding, robotics, segmentation, scene analysis, image representation

**CMU-CS-93-115** **CLASS NOTES: PROGRAMMING PARALLEL ALGORITHMS**
**CS 15-840B (FALL 1992)**
Guy E. Blelloch, Jonathan C. Hardwick
*February 1993*
KEYWORDS: Parallel algorithms, parallel machine models, NESL

**CMU-CS-93-122** **WHAT IS THE CENTER OF THE IMAGE?**
Reg G. Willson, Steven A. Shafer
*April 1993*
KEYWORDS: Image center, camera calibration, automated zoom lens, computer vision, Calibrated Imaging Laboratory

**CMU-CS-93-128** **MOBILE HOST INTERNETWORKING USING IP LOOSE SOURCE ROUTING**
David B. Johnson
*February 1993*
KEYWORDS: Mobile computing, internetworking, mobile hosts, mobile IP, network protocols, routing

**CMU-CS-93-129**[1] **NESL: A NESTED DATA-PARALLEL LANGUAGE (VERSION 2.6)**
Guy E. Blelloch
*April 1993*
KEYWORDS: Data-parallel, parallel algorithms, supercomputers, nested parallelism, PRAM model, parallel programming languages, collection-oriented languages

**CMU-CS-93-130** **SOLVING INTEGER PROGRAMS FROM DEPENDENCE AND SYNCHRONIZATION PROBLEMS**
Jaspal Subhlok
*March 1993*
KEYWORDS: Exact dependence testing, integer programming, parallelizing compilers, parallel program analysis, synchronization analysis

**CMU-CS-93-133** **VISUAL REPRESENTATIONS AS FEEDBACK IN A PROGRAMMABLE VISUAL SHELL**
Francesmary Modugno, Brad A. Myers
*March 1993*
KEYWORDS: User interfaces, intelligent interfaces, end-user programming, programming by demonstration, demonstrational systems, visual languages

---

[1] Revised version of CMU-CS-92-103, January 1992.

CMU-CS-93-134 **TYPED OUTPUT AND PROGRAMMING IN THE INTERFACE**
Francesmary Modugno, Brad A. Myers
*March 1993*
KEYWORDS: User interfaces, intelligent interfaces, end-user programming, programming by demonstration, demonstrational systems, interaction techniques

CMU-CS-93-139 **PREDICTING UNSEEN TRIPHONES WITH SENONES**
Mei-Yuh Hwang, Xuedong Huang, Fileno Alleva
*April 1993*
KEYWORDS: Shared-distribution models, senones, generalized triphones, decision trees, entropy, cross entropy

CMU-CS-93-141 **FULL ABSTRACTION FOR A SHARED VARIABLE PARALLEL LANGUAGE**
Stephen Brookes
*April 1993*
KEYWORDS: Denotational semantics, operational semantics, parallel programming, partial correctness, safety properties, liveness properties, fairness

# TECHNICAL REPORTS 1993

## School of Computer Science

**CMU-CS-93-100** **UNCERTAINTY IN OBJECT POSE DETERMINATION WITH THREE LIGHT-STRIPE RANGE MEASUREMENTS**
Keiichi Kemmotsu, Takeo Kanade
*January 1993*
KEYWORDS: Computer vision, object recognition, interpretation tree, geometric constraints, pose determination, geometric uncertainties

**CMU-CS-93-101** **A METHODOLOGY AND SOFTWARE ENVIRONMENT FOR TESTING PROCESS MODEL'S SEQUENTIAL PREDICTIONS WITH PROTOCOLS**
Frank E. Ritter
Thesis (Psychology)
*December 1992*
KEYWORDS: Spreadsheets, programmer workbench, program editors, tracing, display algorithms, training, help and documentation, model validation and analysis, cognitive simulation, relations among models, protocol analysis, Soar

**CMU-CS-93-102** **EXPERIMENTS IN MULTIPLE-BASELINE STEREO**
Tomoharu Nakahara, Takeo Kanade
*August 1992*
KEYWORDS: Stereo, 3-D vision, range sensing, multi-image fusion

**CMU-CS-93-103** **REINFORCEMENT LEARNING FOR ROBOTS USING NEURAL NETWORKS**
Long-Ji Lin
Thesis
*January 1993*
KEYWORDS: Machine learning, reinforcement learning, artificial neural network, mobile robot, control, credit assignment, temporal difference, action model, teaching, hierarchical learning, abstraction, hidden state

**CMU-CS-93-104** **YARF: AN OPEN-ENDED FRAMEWORK FOR ROBOT ROAD FOLLOWING**
Karl Kluge
*February 1993*
KEYWORDS: Vision and scene understanding, robotics, segmentation, scene analysis, image representation

**CMU-CS-93-105** **NOT AVAILABLE**

**CMU-CS-93-106** **NOT AVAILABLE**

**CMU-CS-93-107** **CRYPTOGRAPHY: IT'S NOT JUST FOR ELECTRONIC MAIL ANYMORE**
J. Doug Tygar, Bennet Yee
*March 1993*
KEYWORDS: Cryptography, franking, electronic currency, mail, postage, stamps, electronic stamps, secure coprocessors, signatures

**CMU-CS-93-108** **THE SECOND GARNET COMPENDIUM: COLLECTED PAPERS 1990-1992**
Brad A. Myers, *editor*
*February 1993*
KEYWORDS: Garnet, user interface development environments, user interface management systems, toolkits, constraints, interface builders, object-oriented programming, direct manipulation

**CMU-CS-93-109** NOT AVAILABLE

**CMU-CS-93-110** NOT AVAILABLE

**CMU-CS-93-111** APPLYING HIGH-LEVEL LANGUAGE PARADIGMS TO COMMUNICATIONS SOFTWARE FOR DISTRIBUTED SYSTEMS
Ellen H. Siegel
Thesis
*January 1993*
KEYWORDS: Distributed systems, communication, high-level programming languages

**CMU-CS-93-112** NOT AVAILABLE

**CMU-CS-93-113** NOT AVAILABLE

**CMU-CS-93-114** NOT AVAILABLE

**CMU-CS-93-115** CLASS NOTES: PROGRAMMING PARALLEL ALGORITHMS
CS 15-840B (FALL 1992)
Guy E. Blelloch, Jonathan C. Hardwick
*February 1993*
KEYWORDS: Parallel algorithms, parallel machine models, NESL

**CMU-CS-93-116** SIGNED VECTOR TIMESTAMPS: A SECURE PROTOCOL FOR PARTIAL ORDER TIME
Sean W. Smith, J. Doug Tygar
*October 1991; version of February 1993*
KEYWORDS: Security, cryptographic controls, distributed systems, concurrency

**CMU-CS-93-117** NEURAL REPRESENTATION OF SPACE USING SINUSOIDAL ARRAYS
David S. Touretzky, A. David Redish, Hank S. Wan
*March 1993*
KEYWORDS: Neural modelling, spatial reasoning, parietal cortex, sinusoidal arrays

**CMU-CS-93-118** NOT AVAILABLE

**CMU-CS-93-119** THE MIDWAY DISTRIBUTED SHARED MEMORY SYSTEM
Brian N. Bershad, Matthew J. Zekauskas, Wayne A. Sawdon
*March 1993*
KEYWORDS: Distributed shared memory, memory consistency, distributed systems, parallelism

**CMU-CS-93-120** NOT AVAILABLE

**CMU-CS-93-121** USING THE MACH COMMUNICATION PRIMITIVES IN X11
Michael Ginsberg, Robert V. Baron, Brian N. Bershad
*March 1993*
KEYWORDS: X11, performance, IPC, shared memory, window systems

**CMU-CS-93-122** WHAT IS THE CENTER OF THE IMAGE?
Reg G. Willson, Steven A. Shafer
*April 1993*
KEYWORDS: Image center, camera calibration, automated zoom lens, computer vision, Calibrated Imaging Laboratory

**CMU-CS-93-123** NOT AVAILABLE

**CMU-CS-93-124** NOT AVAILABLE

CMU-CS-93-125 **THE PRIORITY INVERSION PROBLEM AND REAL-TIME SYMBOLIC MODEL CHECKING**
Sergio V. Campos
*April 1993*
KEYWORDS: Symbolic model checking, real-time systems, priority inversion

CMU-CS-93-126 **NOT AVAILABLE**

CMU-CS-93-127 **NOT AVAILABLE**

CMU-CS-93-128 **MOBILE HOST INTERNETWORKING USING IP LOOSE SOURCE ROUTING**
David B. Johnson
*February 1993*
KEYWORDS: Mobile computing, internetworking, mobile hosts, mobile IP, network protocols, routing

CMU-CS-93-129[1] **NESL: A NESTED DATA-PARALLEL LANGUAGE (VERSION 2.6)**
Guy E. Blelloch
*April 1993*
KEYWORDS: Data-parallel, parallel algorithms, supercomputers, nested parallelism, PRAM model, parallel programming languages, collection-oriented languages

CMU-CS-93-130 **SOLVING INTEGER PROGRAMS FROM DEPENDENCE AND SYNCHRONIZATION PROBLEMS**
Jaspal Subhlok
*March 1993*
KEYWORDS: Exact dependence testing, integer programming, parallelizing compilers, parallel program analysis, synchronization analysis

CMU-CS-93-131 **NOT AVAILABLE**

CMU-CS-93-132 **NOT AVAILABLE**

CMU-CS-93-133 **VISUAL REPRESENTATIONS AS FEEDBACK IN A PROGRAMMABLE VISUAL SHELL**
Francesmary Modugno, Brad A. Myers
*March 1993*
KEYWORDS: User interfaces, intelligent interfaces, end-user programming, programming by demonstration, demonstrational systems, visual languages

CMU-CS-93-134 **TYPED OUTPUT AND PROGRAMMING IN THE INTERFACE**
Francesmary Modugno, Brad A. Myers
*March 1993*
KEYWORDS: User interfaces, intelligent interfaces, end-user programming, programming by demonstration, demonstrational systems, interaction techniques

CMU-CS-93-135 **NOT AVAILABLE**

CMU-CS-93-136 **NOT AVAILABLE**

CMU-CS-93-137 **NOT AVAILABLE**

CMU-CS-93-138 **NOT AVAILABLE**

---

[1]Revised version of CMU-CS-92-103, January 1992.

CMU-CS-93-139 **PREDICTING UNSEEN TRIPHONES WITH SENONES**
Mei-Yuh Hwang, Xuedong Huang, Fileno Alleva
*April 1993*
KEYWORDS: Shared-distribution models, senones, generalized triphones, decision trees, entropy, cross entropy

CMU-CS-93-140 **NOT AVAILABLE**

CMU-CS-93-141 **FULL ABSTRACTION FOR A SHARED VARIABLE PARALLEL LANGUAGE**
Stephen Brookes
*April 1993*
KEYWORDS: Denotational semantics, operational semantics, parallel programming, partial correctness, safety properties, liveness properties, fairness

# TECHNICAL REPORTS 1993

## School of Computer Science

| | |
|---|---|
| **ALLEVA, Fileno** | CMU-CS-93-139 |
| **BARON, Robert V.** | CMU-CS-93-121 |
| **BERSHAD, Brian N.** | CMU-CS-93-119, CMU-CS-93-121 |
| **BLELLOCH, Guy E.** | CMU-CS-93-115, CMU-CS-93-129 |
| **BROOKES, Stephen** | CMU-CS-93-141 |
| **CAMPOS, Sergio V.** | CMU-CS-93-125 |
| **GINSBERG, Michael** | CMU-CS-93-121 |
| **HARDWICK, Jonathan C.** | CMU-CS-93-115 |
| **HUANG, Xuedong** | CMU-CS-93-139 |
| **HWANG, Mei-Yuh** | CMU-CS-93-139 |
| **JOHNSON, David B.** | CMU-CS-93-128 |
| **KANADE, Takeo** | CMU-CS-93-100, CMU-CS-93-102 |
| **KEMMOTSU, Keiichi** | CMU-CS-93-100 |
| **KLUGE, Karl** | CMU-CS-93-104 |
| **LIN, Long-Ji** | CMU-CS-93-103 |
| **MODUGNO, Francesmary** | CMU-CS-93-133, CMU-CS-93-134 |
| **MYERS, Brad A.** | CMU-CS-93-108, CMU-CS-93-133, CMU-CS-93-134 |
| **NAKAHARA, Tomoharu** | CMU-CS-93-102 |
| **REDISH, A. David** | CMU-CS-93-117 |
| **RITTER, Frank E.** | CMU-CS-93-101 |
| **SHAFER, Steven A.** | CMU-CS-93-122 |
| **SAWDON, Wayne A.** | CMU-CS-93-119 |
| **SIEGEL, Ellen H.** | CMU-CS-93-111 |
| **SMITH, Sean W.** | CMU-CS-93-116 |
| **SUBHLOK, Jaspal** | CMU-CS-93-130 |
| **TOURETZKY, David S.** | CMU-CS-93-117 |
| **TYGAR, J. Doug** | CMU-CS-93-107, CMU-CS-93-116 |
| **WAN, Hank S.** | CMU-CS-93-117 |
| **WILLSON, Reg G.** | CMU-CS-93-122 |
| **YEE, Bennet** | CMU-CS-93-107 |
| **ZEKAUSKAS, Matthew J.** | CMU-CS-93-119 |

# Abstract

Over the last half decade, vision based road following systems have progressed from programs which could travel tens of meters between failures to programs capable of driving many kilometers between failures. System performance is typically limited by the following factors: the reliability of the image segmentation techniques used; the accuracy of the system's estimate of road shape and location; the robustness of the system's shape estimation algorithms when faced with data contaminated by bad observations; and the ability of the system to detect and adapt to changes in road structure and appearance.

The YARF road following system presents novel approaches to improving performance in each of these areas. YARF is able to simplify the image segmentation problem by incorporating information about feature appearance as well as feature geometry in the model of road structure. Estimation of the road shape parameters in a data-dependent coordinate system produces dramatic increases in the accuracy of road shape estimation. Use of a robust estimation technique allows YARF to correctly determine the road shape in situations where a least squares based technique would fail due to contaminating data points. Finally, YARF includes techniques for detecting changes in road appearance, verifying intersections or changes in lane structure predicted by a map of the road network, and extracting a model of the visible lane structure of a road from an image. YARF has been tested on a variety of road scenes using a mixture of open- and closed-loop test runs on the Navlab vehicles as well as data collected on videotape and simulations.

# Acknowledgments

I'd like to thank Chuck Thorpe, my thesis advisor, for his help and guidance over the last few years. He was the person who initially steered me towards the topic of vision based road following, and has provided continuous support and advice since then. His availability for technical discussions, enthusiasm about the research, patience in reading and commenting on drafts of the thesis, and willingness to listen when I needed to blow off steam have been much appreciated.

I'd also like to thank Takeo Kanade, who served as my advisor prior to my thesis as well as serving on my thesis committee. He taught me many valuable lessons about performing research and communicating the results clearly and effectively. Thanks also to Tom Mitchell and Ed Riseman, the other members of my committee, for their involvement and comments.

This thesis would not have been possible without the other members of the Unmanned Ground Vehicle project who help keep the vehicle hardware and software up and running. James Frazier, in particular, has risked life and limb serving as the safety driver during YARF's development, and has provided help with the vehicle hardware above and beyond the call of duty. Over the course of this project a number of other people have worked with me on this research. Thad Druffel programmed a number of significant chunks of the system. Didier Aubert developed the initial implementations of a number of the feature trackers used. In addition, Jill Crisman and Tony Stentz contributed code from their research for reuse in YARF.

Additional thanks go to Bill Ross and Jim Moody, who keep the VASC group's computing infrastructure running smoothly. Thanks also to all the people who keep the School of Computer Science functioning. In particular, thanks to Sharon Burks, who has helped guide me through the murky undergrowth of SCS procedure on any number of occasions over the last 7-1/2 years.

On the nontechnical side, thanks to the many great friends I've had here over the years. This includes, among others, the people in the Hill Street Blues viewing group, the poker group, the running and weight lifting group, and the Dinner Co-op.

Last, but not least, thanks go to my sister, Ruth-Margaret, who asked for a baby brother, and to my parents, Richard and Carolyn, who obliged. They have been unfailing in their love and support, and I couldn't have done this without their encouragement over the years.

# Contents

# Contents

# Contents

# 1 Introduction

> "Suppose someone arbitrarily proposes to build a gadget having the following properties: it is to be mounted on an automobile and made to steer the car so that it will follow a white line on the roadway, to slow the car when the line turns blue, and perhaps to perform other bizarre functions. No such device will actually be built, for there is obviously no application that warrants it." ([6], page 72)

When Vannever Bush made that statement in 1949 it is unlikely that he realized the extent to which the private automobile would become a dominant mode of transportation in the United States and elsewhere. Just over 80% of the intercity passenger miles travelled in the U.S. each year are done by private automobile [55]. Over the last decade there has been increasing interest in developing systems capable of autonomous driving in order to increase driver comfort and safety and to increase the volume of traffic that can be carried by existing roadways. Current government projects supporting research into autonomous road following include the DARPA Unmanned Ground Vehicles project [34], the FHWA Intelligent Vehicles and Highway Systems project in the United States, and the Eureka PROMETHEUS project in Europe [24]. Automobile manufacturers such as General Motors [27], Honda [20], Nissan [21], and Daimler-Benz [17] are also performing research in this area.

This thesis describes YARF *(Yet Another Road Follower)*, a vision-based system for autonomous road following. Video data from a camera mounted on a robot vehicle is fed into a computer, which analyzes the image data to locate the position of the vehicle relative to the road. The computer then issues commands to actuators attached to the throttle, brakes, and steering in order to drive the vehicle along the road. YARF has been extensively tested using a combination of open- and closed-loop runs on testbed vehicles, simulation, and data from videotapes. YARF provides a set of perception capabilities to locate the position of the vehicle relative to the road, to detect changes in the lane structure of the road, to navigate through intersections given a model of the intersection geometry, and to extract the lane structure of the road without a prior model.

The central theme of YARF is that using richer models improves road following performance. Models of geometric structure, of road appearance, and of segmentation performance all simplify processing and contribute to improved reliability. YARF uses road models in several ways:

- *Model driven segmentation.* YARF uses model information about feature appearance to select a detection method from a suite of specialized segmentation operators. The simple and fast segmentation techniques used take advantage of model information about feature appearance to robustly detect the presence and absence of different types of features.

- *Exploitation of model coherence to avoid the influence of contaminating data.* The geometric constraints provided by the model are combined with estimation techniques that are robust in the presence of contaminating data observations. This combination permits the system to perform correctly in situations which would cause least squares based approaches to fail.

- *Data driven recognition of model changes.* Failures of the image segmentation techniques to see predicted features are used to trigger hypotheses about changes in road structure. The changes which will occur are assumed known, but the occurrence of the changes is detected in a data driven fashion.

While YARF assumes that the models of road structure used are generated off line, the thesis also presents an algorithm designed to automatically extract much of the needed model information. The algorithm uses a weak domain model to filter a noisy image segmentation, extracting both feature geometry and type.

While the domain chosen for investigating these issues is vision based road following, the techniques developed are applicable to other vision based navigation domains. The characteristics of such domains involve an environment which can be characterized by a geometric model, and the ability to determine the vehicle position relative to the model based on point measurements of the locations of features from the model. The architecture of the YARF system permits easy extension to include additional segmentation or estimation techniques. The object oriented task decomposition built into the system allows simple replacement and extension of functional sets of routines such as those which implement the camera and environment model. Using the system to perform a docking task, for instance, would involve adding detectors to locate features on the docking target, modifying the formulation of the estimation problem to be solved from fitting a road model to fitting the vehicle attitude with respect to the target, and changing the control commands issued. All of the system support in terms of camera modeling, feature detector management, estimation routines, etc. would be available for direct application to the new task.

## 1.1 Why vision-based road following?

One option for autonomous vehicle guidance is a cooperative approach. In such an approach, active or passive beacons are added to the road infrastructure in order to simplify the problem of determining the vehicle location relative to the road. An example of this approach is the lateral control work within the California PATH program [45]. Cheap ceramic permanent magnets are embedded along the center of the lane at one meter intervals. Four Hall-effect magnetometers are mounted under the vehicle bumper, and the field strengths measured by these sensors indicate the lateral offset of the vehicle from the lane center. Such a technique does not require complex perception algorithms, and uses relatively mature technology.

The disadvantage of such techniques is that they require modifications to the road infrastructure. Modifying the entire highway network would take time, and consumers are unlikely to want to pay for expensive options which are useful only in limited areas. The infrastructure modifications would cost money. In addition, closing lanes to embed cables or markers in the pavement would (temporarily) reduce road capacity in the very areas where capacity is a key concern.

The alternative examined in this work is to take advantage of the existing visual cues used by humans to perform lateral vehicle control, i.e. lane markings and pavement edges. This option requires more complicated algorithms and is a less mature technology, but avoids the

policy disadvantages associated with the cooperative approaches [23]. The first system sold will be able to work on existing marked roads, and will require no infrastructure modification (although it may be desirable to repaint lines more frequently).

## 1.2 Perception: an operational level subtask

The driving task has been divided into three levels of subtasks [35]: the *strategic* level, the *tactical* level, and the *operational* level. YARF performs perception tasks which lie at the operational level in this model of driving. Various efforts are under way to develop techniques for performing tasks autonomously at the higher strategic and tactical levels.

Strategic level tasks are global in nature. The primary strategic level task for driving is route selection. This can be based on static map information, or it can combine the static map information with dynamic information about traffic conditions and temporary blockages due to accidents or construction. A number of systems are under development to provide strategic-level driving aids. European efforts are described in [8]. These systems typically include a map database on CD-ROM, a display interface to present the map to a human driver for initial location and destination selection, and sensors (typically wheel encoders and compasses) used to maintain an estimate of vehicle location on the map. The *Travelpilot*$^{TM}$ system developed by Etak and Bosch [7] is an example of the hardware infrastructure available to support such strategic tasks. It includes a CD-ROM map database, processor, display unit, and sensors. While such systems are currently intended as aids for human drivers, they provide a digital map database and vehicle position tracking capability which could be interfaced to provide support for strategic level tasks in an autonomous vehicle.

The tactical level applies domain knowledge (the "rules of the road") and strategic goals to a model of the current environment around the vehicle. Values are chosen for control variables to insure that the vehicle remains in a safe and legal state. The ULYSSES system [42] developed by Reece embodies a detailed model of the tactical level of the driving task. ULYSSES operates within a simulated world, the PHAROS fine-grained traffic simulator. It assumes a set of 14 perception routines such as "find next sign", "find current lane", "mark adjacent lane", "find signal", and "find next overhead sign". In ULYSSES these generate their results by examining the symbolic internal state of the PHAROS simulator. Physical and domain constraints are used to generate bounds on acceptable vehicle acceleration, lane choice, and flags indicating whether the vehicle is in an intersection, and whether it is waiting for right of way. ULYSSES will not violate the lane abstraction in order to avoid a collision. Stengel and Niehaus [46] also constructed an expert system to perform the tactical driving task. Their system detects situations in which a collision appears immanent, and switches to an emergency mode in which collision avoidance is the only goal.

The operational level consists of the low level perceptual and control subtasks which provide the support for the strategic and tactical levels. Reece identified 14 perceptual routines needed to support his tactical model of driving ([42], pages 170-172). Those which relate to detecting the road location and structure are:

> *Find current lane*. Locate the boundaries of the lane the vehicle is in, determining the heading and offset of the vehicle with respect to the lane.

*Mark adjacent lane.* Check for lane boundaries near a marked location in the image.

*Track lane.* Start at a marked location in a lane and track the lane as far as possible in the image.

*Profile road.* Identify the lane structure of the road, reporting the location and type of lane markings.

*Find intersection roads.* Locate all the approach roads and lanes at an intersection.

*Find path in intersection.* Similar to "find intersection roads", but selects a lane to turn from and generates a path through the intersection.

YARF maintains a model of the current lane structure and road curvature, and an estimate of the location of the vehicle relative to the road. Given these, YARF provides the capability of "find current lane". It applies perception to verify the model's information about other lanes, providing the capability of "mark adjacent lane" and "track lane". The SHIVA algorithm described in Chapter 6 extracts a symbolic description of lane structure from an image, implementing the capability of "profile road". YARF uses a combination of perceptual cues and map information to recognize the approach of an intersection, verify the location of the destination lane across the intersection, and plan a path to navigate through the intersection. This implements most of the capability of "find path in intersection".

YARF's symbolic model of road structure contains the information necessary to constrain the search for other traffic objects used in the ULYSSES model such as signs adjacent to the road, overhead signs, and cars in a specified lane. The YARF system is thus extensible to include other primitive perception routines identified by Reece, but not currently implemented within the YARF system.

## 1.3 The centrality of the symbolic road model

The model of road structure and appearance, whether implicit or explicit, lies at the core of a road-following system. Models which explicitly represent information about feature location and appearance can simplify the image segmentation problem faced by the system. Models which make explicit the geometry of the lane structure of the road can use that information to take advantage of redundancy in the data and inter-feature constraints. Explicitly detecting and representing locations where perception fails to verify the expectations generated by the model allows a system to detect changes in road appearance and react appropriately. The family of road models used generates constraints which can be used to filter a noisy image segmentation and extract a symbolic model of the visible lane structure. YARF takes advantage of each of these possibilities.

## 1.4 Experimental testing of YARF

The YARF system has been tested on the Navlab I and Navlab II testbed vehicles developed at Carnegie Mellon. These vehicles have served as platforms for investigating a variety of issues in outdoor autonomous navigation [52] [53] [49]. Navlab I is a modified Chevy van

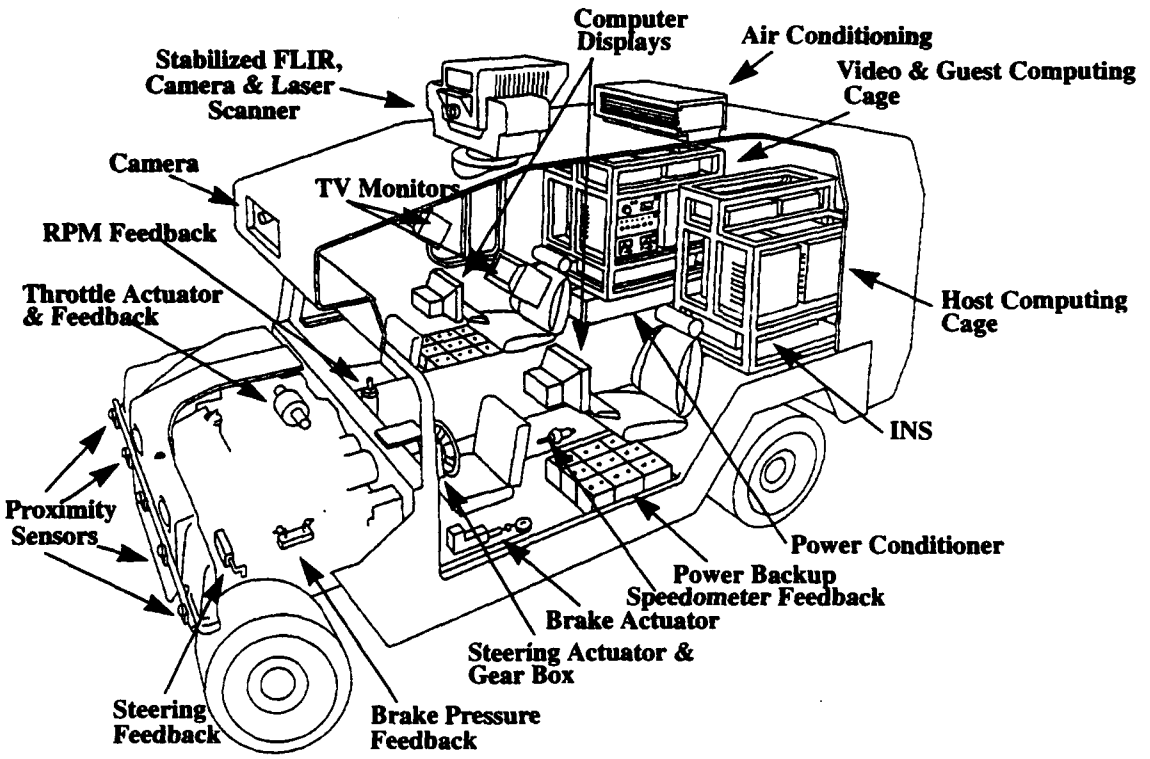



**Figure 3: Cutaway view of Navlab II interior**

YARF uses a predict-verify-update technique to track the road. The estimated road shape from the previous image is transformed using an estimate of the vehicle motion between images to predict the location of the road in the current image. Image segmentation techniques are applied in small search windows around the expected feature locations to verify the actual feature locations. These new data points are then used to update the estimated road shape. Previous systems have used a single segmentation technique for road tracking, however this is inadequate. Pavement edges, white painted stripes, and yellow painted stripes are not equally detectable by a single segmentation technique due to differences in contrast and boundary sharpness. The YARF system provides an extensible collection of simple segmentation techniques, each of which is designed to reliably detect the presence or absence of a particular type of road feature. These segmentation techniques, referred to as *feature trackers*, are described in Chapter 3.

YARF uses statistical estimation techniques to combine the individual point feature locations detected by the feature trackers into an estimate of the road shape parameters. YARF, like a number of other road followers, estimates the shape parameters of a non-linear model from the coefficients of a polynomial approximation based on a series expansion. Chapter 4 demonstrates that performing the parameter fitting in a data-dependent coordinate system reduces the errors introduced by this approximation by a factor of 5-10 for roads with large curvatures. In addition to this improvement in performance, YARF also takes a unique approach in its selection of estimation technique. Systems which use a least-squares based approach are vulnerable to failures caused by false feature data points. YARF is the first road following system to use LMS (*Least Median of Squares*) estimation to avoid such

failures. LMS uses a criterion function which selects a parameter estimate based on the internal consistency of the valid data within a data set containing contaminating false data points.

The basic road tracking loop described in Chapters 3 and 4 assumes that the road has a constant lane structure, with features separated by constant widths. In an urban driving environment, road features may disappear or change location due to approaching intersections or changes in lane structure. As a result, for safety reasons it is important that an autonomous road follower detect situations where its model of the road no longer appears to reflect reality. Such situations may be expected and represented in the mission map, or they may be unexpected. Chapter 5 describes the techniques YARF uses to react to expected changes in road appearance. YARF uses a map of the roads to be traversed in the course of a mission. This map is generated off line. By using a data-driven algorithm for detecting the approach of changes in lane structure YARF eliminates the need for a precise metric model of the road network. Local geometry at an intersection is represented, but distances and road curvature between intersections are not represented. YARF determines the position of the vehicle relative to the local intersection coordinate system based on image information about the termination of features as the vehicle approaches the intersection. Currently the mission map is static, and is not updated to reflect information gathered in the course of performing missions using the map. One direction for future work is to eliminate the need for a quantitative representation of intersection geometry, with the goal of building systems which can follow the kind of qualitative and partial driving directions which humans give each other. Another direction for future work is to incorporate learning from experience, so that the system can improve performance on subsequent retraverses of the same route.

One longer term goal of the research agenda set out in this thesis is reducing the dependence of YARF on map information generated off line. The SHIVA algorithm, described in Chapter 6, provides a robust technique for extracting the visible lane structure from an image of the road. Constraints from a simple domain model are used to filter a noisy edge segmentation in order to extract feature geometry. YARF extends the capability provided by similar algorithms by extracting feature type as well as geometry. By applying the specialized feature trackers used in YARF, it is also possible to classify the detected features into pavement edges, white stripes, and yellow stripes. The SHIVA algorithm provides a method for initializing YARF's estimate of the vehicle location and road curvature. It also forms a basis for further research towards the goal of reacting to unexpected changes in the lane structure of the road. Chapter 7 summarizes the contributions of YARF, and describes directions for future work.

# 2 Road modeling in YARF

## 2.1 The role of the road model

The model of road structure and geometry is the central data structure in an autonomous road following system. It provides information needed to guide and interpret the results provided by segmentation of the visual input data, and stores information about the road needed to make tactical driving decisions about lane choice. The limitations of the road model used in a system play a strong role in defining the performance limitations of the system as a whole.

The road scene shown in Figure 4 illustrates several of the roles played by the road model. The vehicle is stopped at a traffic light (not visible). If the route plan indicates that the vehicle should turn left at the upcoming intersection, then the tactical level needs to obtain certain pieces of information from the road model in order to carry out the turn. The road model needs to represent the geometric knowledge that there are three lanes, and that the vehicle is currently in the right lane. It also needs to represent the semantic information that the left edge of the right lane is a white line, while the left edge of the middle lane is a double yellow line, indicating that the vehicle should move one lane to the left in order to make the required turn.

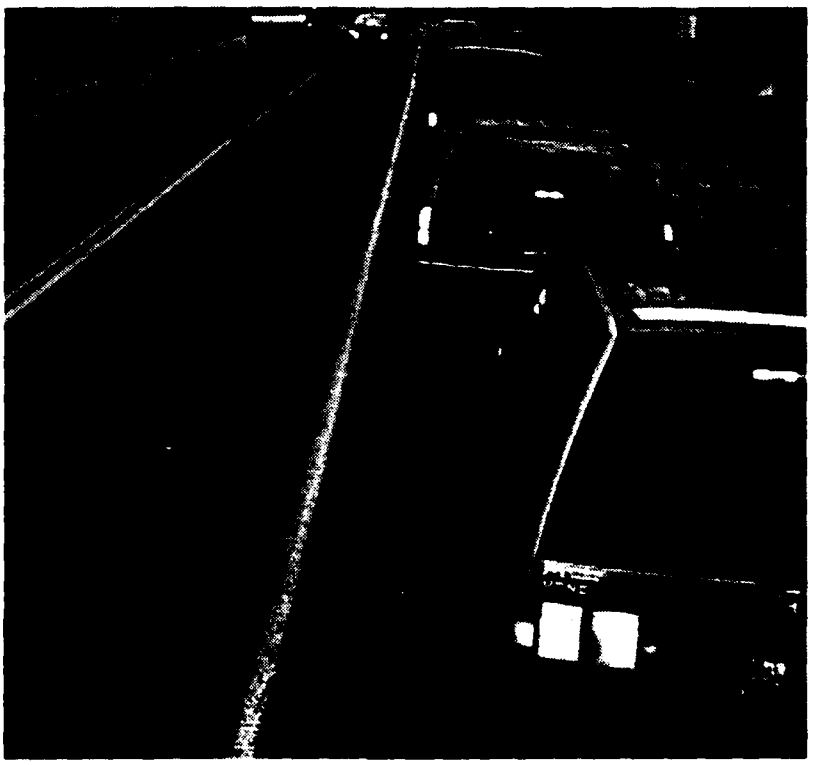


**Figure 4: Road scene**

The operational level also makes demands on the road model. An example of this would be the estimation of camera tilt in each image frame, as is done in the VITA system[18]. Estimation of the tilt requires data from multiple features in order to disambiguate the effects of camera tilt and road orientation. As a result, if the road model includes only the edges of the lane the vehicle is in, then the system will not be able to estimate the tilt for this image due to the occlusion of the entire right edge of the lane. A road model which includes the full lane structure of the road permits the detection of other features in the case that one of the lane edges is obscured.

This chapter presents the concept of *generalized stripes* as a method of road representation. Generalized stripes include information about road feature type and appearance, unlike earlier road representation schemes which only included geometric information. In addition, generalized stripes provide a unifying framework for classifying different representations of road geometry. Generalized stripes are a generic differential representation of road geometry which permit arbitrary terrain variations and banking of the road surface. Previous representations of road geometry can be classified as special cases of generalized stripes. The chapter closes with the mathematical definition of the special case of generalized stripe used in the YARF system. These stripes assume a flat road with no bank and circular arc spine curves.

## 2.2 Generalized stripe road models

The representation chosen for roads in the YARF system is as a sequence of *generalized stripes*. Generalized stripes define a set of surfaces in a manner analogous to the way in which generalized cylinders define a set of volumes. A generalized stripe consists of a one-dimensional feature cross section which is swept perpendicular to a spine curve to define the road surface (see Figure 5). The generalized stripe model of road structure encodes both the semantic information needed at the tactical level ("the left edge of the current lane is delimited by a broken white stripe") and the geometric information needed to perform operational level perceptual tasks.

The generic definition of a generalized stripe is straightforward. The spine curve is defined by the vector function $\vec{S}(u) = \left[x(u)\ y(u)\ z(u)\right]$ where $u$ is the arc length along the spine curve. The bank angle of the road cross section at arc length $u$ along the spine curve is given by the banking function $\beta(u)$. The feature cross section function, $F(v)$, indicates the feature type at offset $v$ from the spine curve. Generalized stripes are distinguished from *ribbons* [41] in two ways. First, they have internal structure corresponding to the road markings. Second, they are not restricted to a plane. The spine of a generalized stripe can be a curve in three-space, and the stripe may bank.

The location of the point at offset $v$ from the spine curve at arc length $u$ along the spine curve is determined as follows. Let $S_u$ be the point on the spine curve at distance $u$ along the spine. Let $\hat{V}$ be a unit vector in the vertical direction at $S_u$, and let $\hat{T}$ be the unit tangent vector to the spine curve at $S_u$. The point at offset $v$ on the feature cross section ($R_{u,v}$) is determined by the following constraints:

- The distance from $S_u$ to $R_{u,v}$ (the length of vector $\vec{P}$ in Figure 6) is equal to $|v|$, the

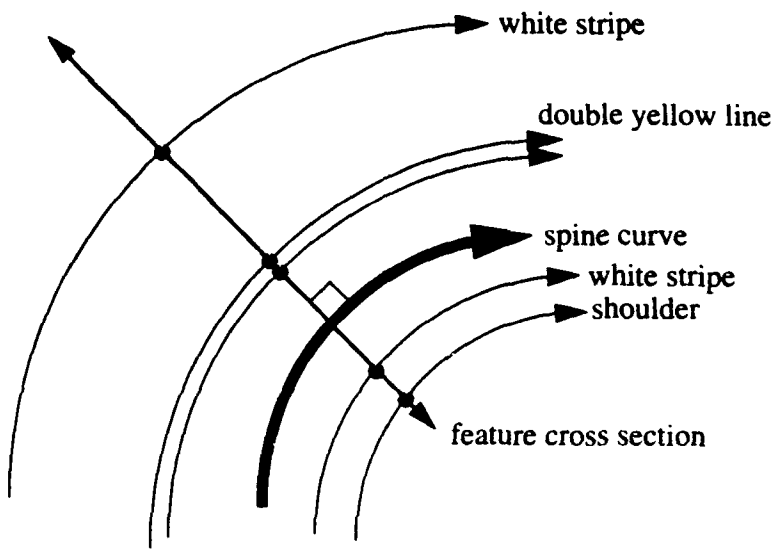



**Figure 5: Generation of a road by sweeping a feature cross section perpendicular to a spine curve**

absolute value of the offset of the point from the spine.

- $\vec{P} \bullet \vec{T} = 0$ (the cross section is swept perpendicular to the spine curve).
- $\vec{P} \bullet \vec{V} = |v| \cos(\frac{\pi}{2} - \beta(u))$ (the cross section has bank $\beta(u)$ at arc length $u$ along the spine curve).
- $\vec{P} \bullet (\vec{T} \times \vec{V})$ has the same sign as $v$ (positive offsets lie to the right of the plane defined by $\vec{V}$ and $\vec{T}$ when facing along $\vec{T}$).

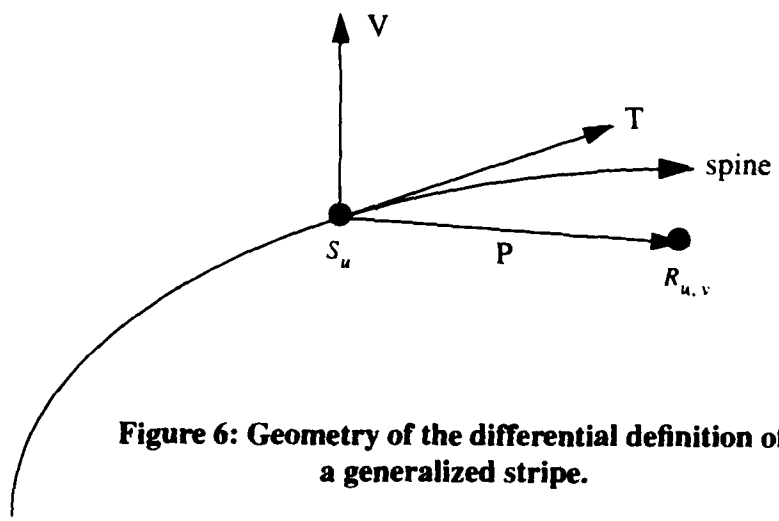



**Figure 6: Geometry of the differential definition of a generalized stripe.**

## 2.3 Previous road models as generalized stripes

Almost all previous schemes for representing road geometry can be viewed as special cases of the generalized stripe abstraction defined above. Generalized stripes in which the bank angle is zero everywhere along the spine ($\beta(u) = 0$) correspond to the zero-bank model described by DeMenthon and Davis [11] and Kanatani and Watanabe [25]. In these models the spine curve is an arbitrary curve in 3-space. These models do not constrain the shape of the spine curve. As a result, the assumption of constant road width used in the algorithms leads to errors in the reconstructed road shape when there are errors in the location of the edges of the road.

Dickmanns and Mysliwetz [13] assume a zero-bank road model, but also constrain the spine curve $\hat{S}(u) = \left[x(u)\ y(u)\ z(u)\right]$ to have a form such that the curves $\hat{S}_H(u) = \left[x(u)\ y(u)\ 0\right]$ and $\hat{S}_V(u) = \left[0\ y(u)\ z(u)\right]$ are clothoids. They use state space estimation techniques to recover the horizontal and vertical curvatures and curvature derivatives of the spine curve. These constraints on the shape of the spine curve make the spine shape estimation much more robust in the presence of noise than the general zero-bank model.

The form of the spine can be restricted further by assuming that the road is confined to a flat ground plane (corresponding to spines of the form $\hat{S}(u) = \left[x(u)\ y(u)\ 0\right]$, with the zero-bank constraint $\beta(u) = 0$). Franke [18] describes such a model in which the spine is constrained to be a clothoid in the ground plane. Earlier work on the VaMoRs project described by Dickmanns and Graefe [12] also used a flat earth clothoid model of the road spine.

The shape of the spine curve can be restricted even further by assuming the spine to be a circular arc (a clothoid with constant curvature) in addition to the flat earth restrictions. A very early version of VaMoRs described by Mysliwetz and Dickmanns [37] used such a model. Schaaser and Thomas describe the use of this type of model in a road following system developed at the University of Bristol [44].

The most restricted form of generalized stripe used in previous road following systems assumed the flat earth constraints and modeled the spine as linear or piecewise linear. Such models were used in the SCARF system [10], the VITS system [54], the LANELOK system [29], and road following work at FMC [31] and the University of Michigan [33] [39].

## 2.4 Generalized stripes in YARF

The current implementation of YARF restricts the spine curve to the flat earth, circular arc case described above. The local road geometry is defined by three parameters. These are the spine curvature, spine $x$ intercept, and spine tangent at its $x$ intercept, expressed in a vehicle-centered coordinate system (see Figure 7).

Such a model of road geometry limits the ability of the system to navigate roads with significant deviations from the flat earth assumption, but provides a model with a small number of parameters capable of driving many city roads and highways. Limiting the form of the spine curve to a parametric family of curves with a small number of parameters makes the shape recovery less sensitive to noise in the feature positions extracted from the image

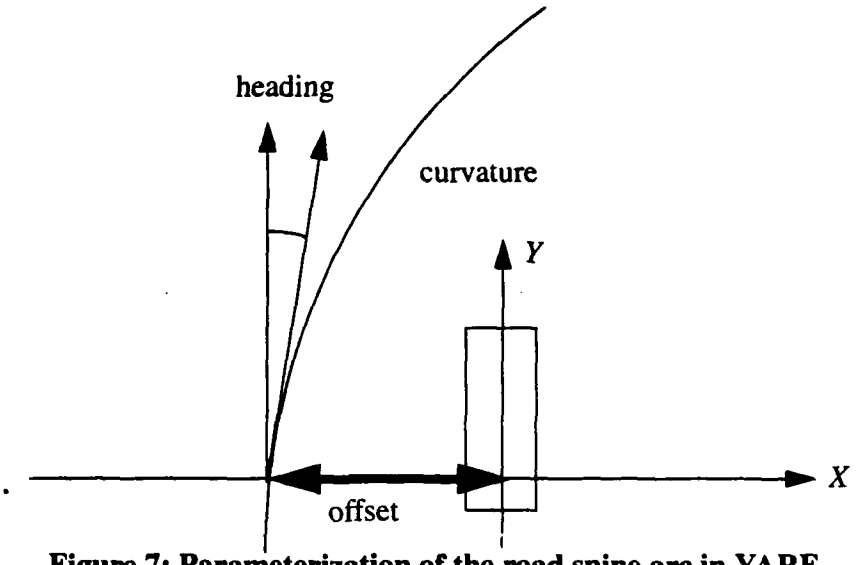



**Figure 7: Parameterization of the road spine arc in YARF**

data. The impiementation of YARF was done with an object-oriented programming style. Thus, replacing the current model with a more general model would involve limited changes restricted to the routines which predict feature location and estimate the road shape parameters.

Coordinate systems in the ground plane are defined with positive curvatures indicating curves to the right and negative curvatures indicating curves to the left. Angles are measured with respect to the *y* axis, with positive angles to the left of the *y* axis and negative angles to the right of the *y* axis (see Figure 8).

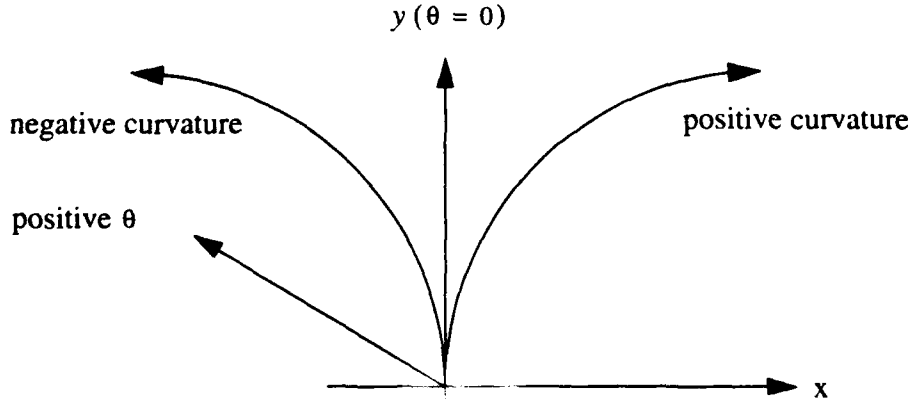



**Figure 8: Conventions for ground plane coordinate systems**

There are two basic predictive procedures associated with the class of generalized stripes used in YARF. The first involves computing the parameters of an arc which is offset from the spine by a given amount. The spine is defined by a point on the spine $(x_S, y_S)$, the tangent direction $\theta_S$ at that point, and the spine curvature $k_S$. The feature has offset $d_F$ from the spine. The procedure returns a point $(x_F, y_F)$ on the feature arc, the tangent $\theta_F$ at that

point, and the curvature $k_F$ of the feature arc. Since curvature is 1 / radius, the feature arc curvature is $k_F = 1/(1/k_S + d_F) = 1/((1 + k_S d_F)/k_S) = k_S/(1 + k_S d_F)$. $(x_F, y_F)$ will be chosen so that it lies radially out from $(x_S, y_S)$, so $(x_F, y_F) = (x_S + d_F \cos\theta_S, y_S + d_F \sin\theta_F)$ and $\theta_F = \theta_S$ (see Figure 9).

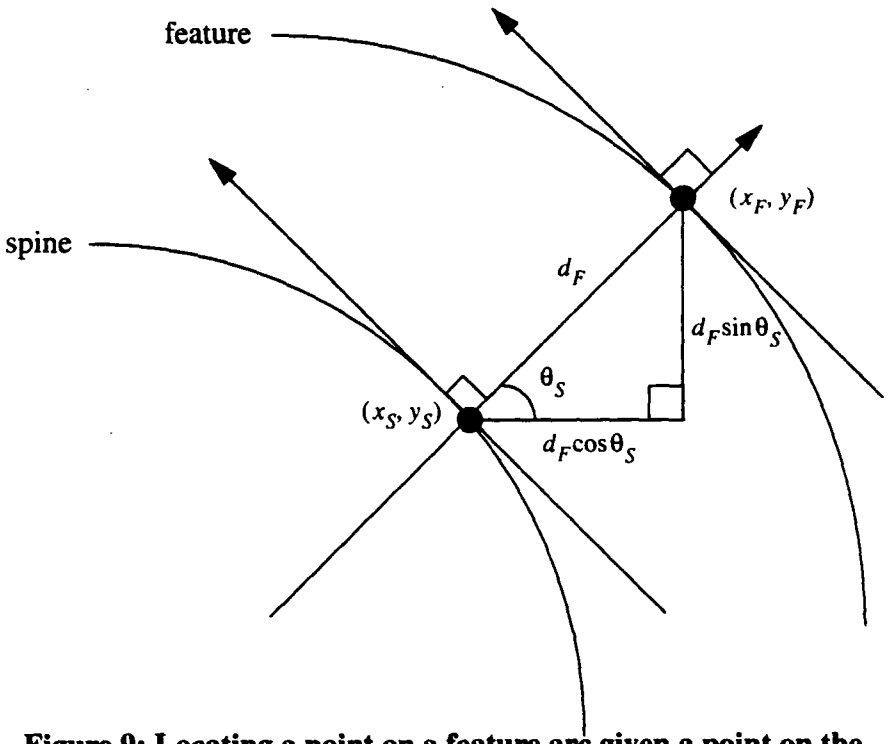



**Figure 9: Locating a point on a feature arc given a point on the spine, the spine tangent at that point, and the spine curvature**

The other basic predictive function returns the $x$ value $(x_f)$ and tangent direction $(\theta_f)$ of the closest point on an arc (if any) with a specified $y$ value. This procedure takes as inputs a circular arc and a $y$ value, $y_f$. The arc is defined by a point on the arc $(x_A, y_A)$, the tangent direction at that point $\theta_A$, and the arc curvature $k_A$. In the case where the curvature is not close to zero, the procedure begins by computing the center of the arc, $(x_C, y_C) = (x_A + (\cos\theta_A)/k_A, y_A + (\sin\theta_A)/k_A)$. The solutions for $x_f$ are then the solutions to the quadratic equation $(x_f - x_C)^2 + [(y_f - y_C)^2 - k_A^{-2}] = 0$. If the quadratic equation has no real solutions, the routine returns an error value indicating that there is no point on the specified arc at that $y$ value. Otherwise, the value for $x_f$ which gives the smaller distance from the point $(x_A, y_A)$ is chosen. The tangent at that point is computed by the formula $\theta_f = \text{atan}((x_C - x_f)/(y_f - y_C)) - (\pi/2)$. YARF uses a convention that the tangent points forward along the arc. The angle computed by the formula is checked to see if it meets that condition. If it does not, $\pi$ is added to $\theta_f$ so that it points in the correct direction (see Figure 10).

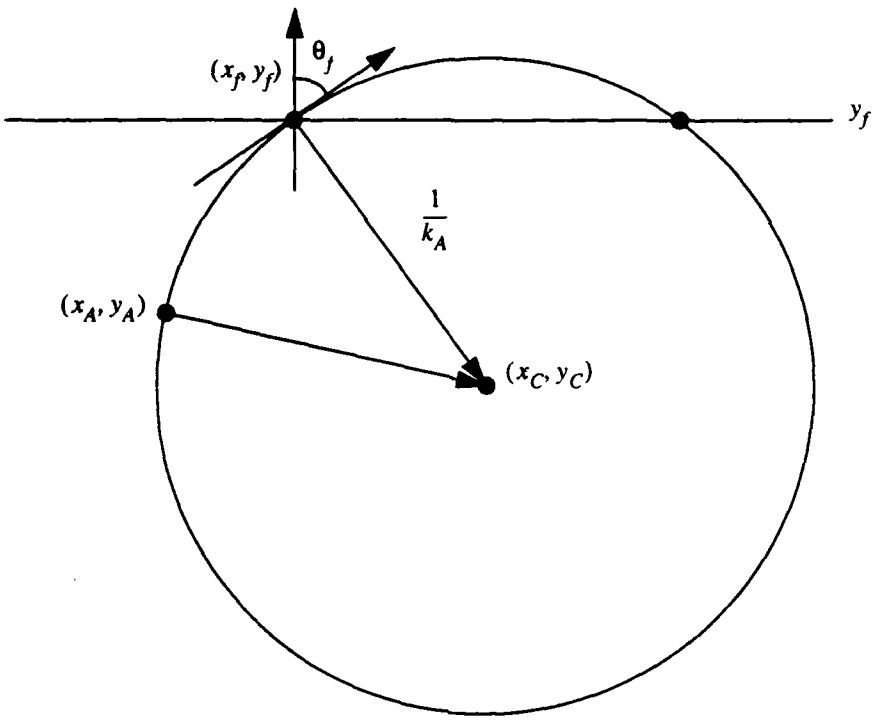



**Figure 10: Geometry of predicting the $x$ value corresponding to a given $y$ value on an arc**

In the case where the curvature is very close to zero, the arc is treated as a straight line. If $\cos\theta_A$ is very close to zero, the line is parallel to the $x$ axis, and the routine returns an error value indicating that there isn't a solution (unless $y_A = y_f$, in which case there are an infinite number of solutions). Otherwise, $\theta_f = \theta_A$ and $x_f = x_A - (y_f - y_A)\tan\theta_A$.

## 2.5 Conclusion

The generalized stripe concept provides a general framework within which different schemes for the representation of road geometry can be classified as special cases which restrict the form of the spine curve and bank function. YARF chooses a form of generalized stripe in which there is no bank to the road and the spine arc is a circular arc confined to a horizontal ground plane. The model of road appearance and geometry is central to YARF, serving a number of functions.

First, it constrains the location, orientation, and appearance of the road features which the system is tracking in the image. Modeling the expected location and type of the road features simplifies the segmentation part of the task in several ways. Knowledge of the feature type being looked for permits the use of specialized segmentation techniques tuned to detect a particular type of feature rapidly and reliably. Knowledge of the expected feature location reduces the number of pixels which need to be examined in an image, and reduces the likelihood of false feature detections by limiting the portion of the image examined.

Second, it provides the constraints relating individual measurements of feature location to the position of the vehicle on the road and the local road curvature. These constraints generate the system of equations which are solved through least squares or least median squares techniques to estimate the vehicle position and road curvature in each frame.

Third, it provides the context for analyzing situations in which expected features were not detected in the image data. By combining map and image data in this way YARF is able to detect and navigate through expected changes in lane structure and intersections.

Each of these functions serves as the theme of one of the remaining technical chapters in the thesis. The final technical chapter deals with the issue of autonomously extracting the feature cross section of a generalized stripe from image data. This is done by using a weak domain model to filter noisy edge data. Such an algorithm can serve several purposes in an autonomous road following system. First, it can locate the road to initialize the system's estimate of the vehicle's location. Second, it can serve as a recovery mechanism when the road structure changes in a way not predicted by the system's existing model of the road network.

# 3 Feature Detection

## 3.1 Introduction

The performance of high level vision systems is ultimately limited by the quality of the low level segmentation algorithms used in the system. Binford [4] observed that this was the case for existing object recognition systems and suggested increased emphasis on research into low level image segmentation. While this is a worthy goal, it does not solve the problem for researchers who wish to build high level vision applications now. Road following systems have also fallen prey to limitations imposed by available low level segmentation techniques. One early system developed at CMU looked for strong edges in the image corresponding to the edges of the road. Unfortunately, the test road had a tree located next to a sharp curve. The system would detect the clean, sharp edges of the tree in the image, and observe that they were a better extension of the road edges in the lower part of the image than the actual road edges as the road curved off. The result was a test run in which the Terregator test vehicle attempted to climb the tree.

YARF is able to use its explicit models of road geometry and appearance to overcome many of the segmentation-related limitations of pervious road following systems. Knowledge of the road geometry allows the system to focus algorithms and computational resources on areas of the image which are likely to contain the features the system is looking for. This reduces both the computational cost of verifying the location of the road and the possibility of false feature detections due to imperfect heuristic segmentation algorithms. YARF explicitly represents the different types of features which compose the road, such as painted stripes of different colors and pavement/terrain boundaries. This allows the system to use multiple segmentation techniques which are based on small models of the expected appearance of particular types of features rather than trying to develop a sufficiently competent general purpose segmentation technique to detect and classify all possible types of road feature with equal reliability.

## 3.2 Previous approaches to road feature extraction

Other road following systems have relied on a single segmentation technique, and can be split into three groups based on the type of information they use to detect road features: edge based, pixel color based, and connectionist. Each of these approaches has limitations which precludes coping with the full range of variability to be encountered in the road following domain.

Edge based systems can be split into those which use a traditional edge operator (typically the Sobel) and those which use various kinds of oriented edge detectors. Systems which locate boundaries of road features using the Sobel edge detector and Hough line extraction include LANELOK [27], MARF [57], and work done at the University of Michigan [39]. Systems which use custom oriented edge detectors include the VaMoRs [12] and VITA [17] systems. Edge detection as a strategy for road feature location suffers from problems caused by extraneous edges due to shadows, puddles, cracks in the pavement, and other blemishes on the road surface. Yellow lines typically have much lower contrast than white painted

lines, making them hard to detect in an intensity image [27]. Edge operators which examine a single band image cannot distinguish between markings of different colors, an ability which is necessary given the semantic meaning of the marking colors.

Systems which use pixel color to detect road features can be divided into systems which use thresholding of a single color feature, systems which classify pixels based on estimates of probability of color class membership, and systems which use domain-specific heuristic region extraction techniques. Systems which extract the road region by thresholding a preselected color feature include the Sidewalk II system[19] and the VITS system[54]. The Sidewalk II system created a histogram of the blue band of a color image. The "best" valley in the histogram was selected as the threshold separating road pixels from nonroad pixels. This simple technique worked adequately in the domain of the system, navigating around a network of paved campus sidewalks bordered by grass. The VITS system used a (blue - red) color feature for segmentation. Rather than have a single threshold separating road and nonroad, the VITS system had an upper and lower bound for pixels which were sunlit road, and an upper and lower bound for pixels which were shadowed road.

Systems which classify pixels based on an estimate of the probability of a given RGB value being road or non-road include the FMC system [31] and the SCARF system [10]. In the FMC system the RGB image values were projected through a linear transform onto a single color feature. Normalized histograms of road and non-road regions in the image are used to adaptively compute the likelihood ratio that a pixel with a given feature value is a road pixel. The SCARF system used Bayesian classification assuming Gaussian color classes whose means and covariances were adaptively determined.

These systems all limit themselves to distinguishing between the road surface and the surrounding off-road terrain, ignoring any internal markings and structure possessed by the road. Region segmentation and pixel classification techniques tend to oversegment or undersegment (often in different parts of the same image), making the extraction of the road region and associated markings difficult. Misclassified pixels result in a noisy segmentation, requiring the use of a robust technique to extract road position and shape. The SCARF system, for instance, assumes the road is locally close to straight and parameterizes each possible road by the angle it makes with vertical in the image and the vanishing column of the road on the horizon row of the image. The road position is determined by a voting scheme in which each road pixel votes for every set of road parameters for which that pixel would lie on the road and against every set of road parameters for which that pixel would lie off the road. Similarly, non-road pixels vote against sets of road parameters which would place them in the road, and for sets of road parameters which would place them off the road surface.

Systems which use a domain specific heuristic technique to extract regions corresponding to white road markings include the University of Bristol system [44] and the Fujitsu system [38]. In the University of Bristol system a pixel is classified as stripe if it is brighter than a threshold intensity level and between edges of opposite sign separated by the expected stripe width. Connected components are extracted to locate possible stripe regions, and arcs are fit to stripe regions to determine the road geometry. In the Fujitsu system, pixels above an adaptively selected intensity threshold are classified as stripe, and a search is done for likely road stripe regions. These systems also fail to distinguish between white and yellow painted stripes, and face the problems caused by the difference in contrast between the two types of painted line.

The ALVINN system [40] explores a connectionist alternative to road detection. Rather than attempt to formulate a segmentation technique to detect various types of road features, a three layer backpropagation network is trained by watching a human drive along a stretch of road for several minutes. The hidden units in the network develop feature detectors for that particular road. The system has driven on roads which range from unpaved, poorly defined dirt roads to well marked multilane highways. The advantage of this approach is that the simple architecture used appears to be able to extract relevant features for a broad variety of roads. The disadvantage is that the system has no model of the road structure, and as a result changes in lane structure result in the network failing to give a reliable steering direction. While such failures can be detected automatically and control returned to a human driver, the only way to recover is to retrain the network.

All these approaches share certain common limitations. They fail to model the internal structure of the road in full detail. Driving requires knowing the color, continuity, and relative location of the markings on the road. Autonomous road following systems, therefore, need to have segmentation techniques which are capable of distinguishing between different color markings and between continuous and broken markings.

One approach to overcoming these limitations would be to search for a "silver bullet" segmentation technique. Such a segmentation technique would extract different road features such as painted lines (white and yellow, single and double, continuous and broken) and pavement/shoulder boundaries with equal reliability. It would also provide a sufficiently rich representation of the segmented image to allow discrimination between these different types of road feature. YARF rejects this approach. No single heuristic based approach to segmentation appears to be robust enough to deal with the full range of variation encountered, while techniques based in physical models of image formation are not mature enough to deal with the unconstrained lighting, weather effects, and mix of natural and synthetic surfaces found in the outdoor environment.

Instead, YARF incorporates multiple simple segmentation techniques specialized to detect different road features and capable of detecting when they have failed to see the expected feature. Such an approach allows the system to discriminate between markings of different colors, and to detect when the markings are broken rather than continuous based on periodic absences of the feature.

The initial implementation of YARF used a variety of feature detection techniques which had been developed in previous road following work[30]. As work progressed, part of the research effort focused on developing robust feature detection algorithms (referred to as *feature trackers* in the remainder of the thesis). This resulted in specialized techniques for detecting yellow and white stripes [1] and pavement boundaries. The next section discusses the feature trackers developed for the YARF system. This is followed by a description of the instrumentation in YARF for evaluating the performance of the feature trackers.

## 3.3 Feature trackers

### 3.3.1 Tracking yellow painted stripes

The ideal feature tracker would exploit a feature which is invariant under a broad variety of environmental conditions. By switching from the RGB color space to the IHS (intensity, hue, saturation) color space the perceived color of a pixel (the hue) can be separated out from the relative brightness (intensity) and whiteness (saturation) of the pixel. Hue appears to provide an invariant feature for detecting yellow painted stripes. Figure 11 illustrates this invariance. The histograms show number of pixels versus hue angle for windows containing a yellow stripe and surrounding pavement. Red is located at 0 degrees on the wheel, green at 120 degrees, and blue at 240 degrees. The top histogram is for an image taken in sunlight with dry pavement. The middle histogram is for an image taken in dense mottled shadows. The bottom histogram is for an image taken on an overcast day with wet pavement. The peak on the left of each histogram consists of pixels from the yellow stripe, while the peak on the right consists of pavement pixels. Note that while the exact location of the yellow stripe peak shifts slightly, it stays firmly within the 40 to 90 degree range of the hue wheel. While the hue of yellow painted stripes appears to be invariant over a broad range of weather conditions, it needs to be noted that the location of the hue ranges for yellow stripe and pavement pixels is dependent on hardware factors such as filters used and the color balance of the individual camera. The library of functions providing the tracker support includes routines to allow the user to examine hue and saturation profiles for windows within an image to test different filter and color balance arrangements.

A feature tracker was developed to exploit this invariance. Pixels in the feature prediction window are classified as yellow stripe or background based on hue and saturation (see Figure 12). Next, a single pass of thinning is done to eliminate isolated yellow stripe pixels. The predicted stripe width must exceed a specified threshold for this thinning to occur in order to avoid thinning the stripe away in windows near the top of an image. The centroid of the pixels classified as yellow stripe is returned as the feature location. If there were no pixels classified as yellow stripe in the window, then the tracker returns an error code to indicate that the feature was not found. The tracker also counts the number pixels in the search window which are black (intensity < 2) or white (intensity > 253). If more than 90% of the pixels in the window are black or white, the tracker returns an error value indicating that the window was saturated due to dynamic range limitations of the camera.

The hue of a pixel is defined as $hue = \operatorname{acos}\left(\frac{2\times R - B - G}{2\times\sqrt{(R-G)^2 + (R-B)\times(G-B)}}\right)$

If $B > G$ then $hue = 2\times\pi - hue$[2]. Colors with a specified hue lie on a plane which contains the intensity axis. This gives two points which lie in that plane, (0, 0, 0) and (255, 255, 255). For colors whose hue falls between red and green (0 and 120 degrees on the color wheel), a third point in that plane can be found by fixing the blue and green components of a pixel and solving the hue equation for the red value.

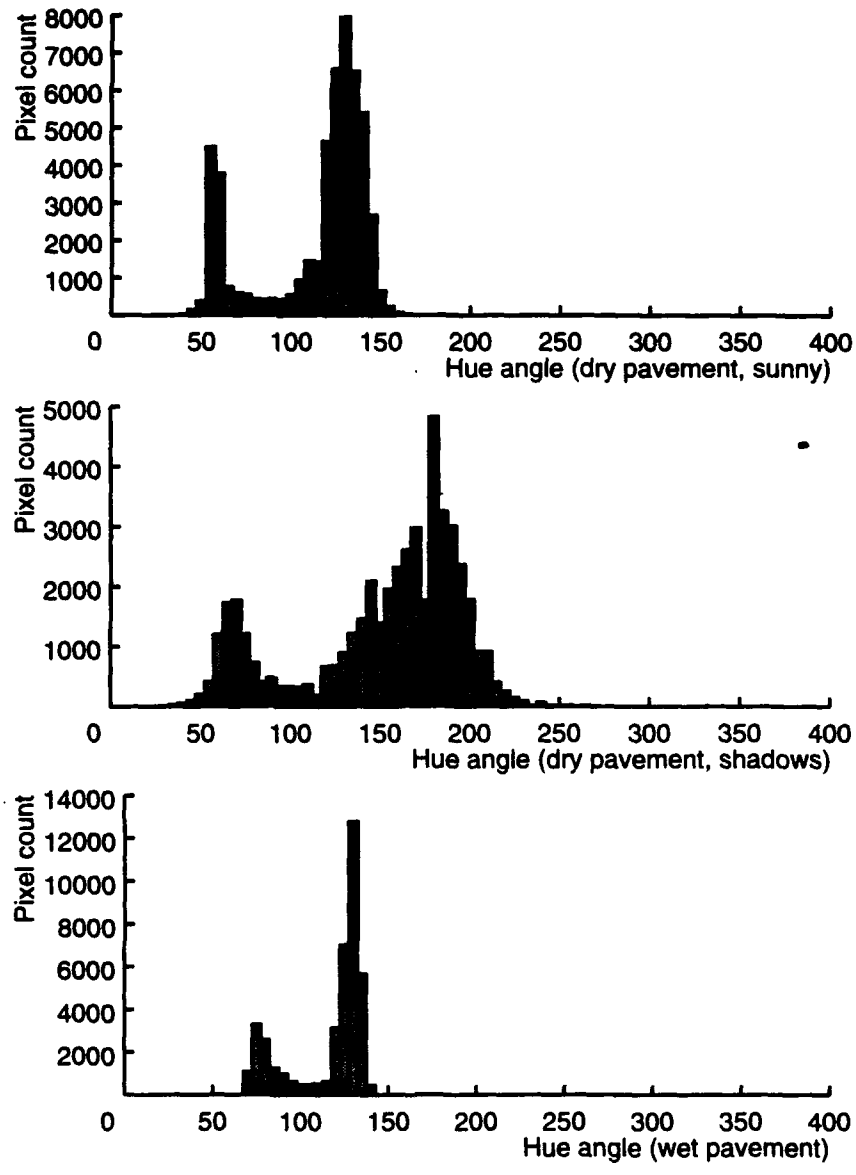



**Figure 11: Stability of yellow stripe hue under different conditions**

Fixing $B = 0$ and $G = 1$, the $R$ value corresponding to a given hue value ( $0 < hue \leq 120$ ) is $R_{hue} = 0.5 + 0.5 \times \sqrt{3} \times (\cos(hue)) / (\sin(hue))$. Given the point ($R_{hue}$, 1, 0) with hue value *hue*, the normal to the plane containing color values which have that hue is $\left[R_{hue}\ 1\ 0\right] \times \left[1\ 1\ 1\right] = \left[1\ (-R_{hue})\ (R_{hue} - 1)\right]$. The equation of the plane is therefore $R - R_{hue} \times G + (R_{hue} - 1) \times B = 0$, which simplifies to $(R - B) + (B - G) \times R_{hue} = 0$.

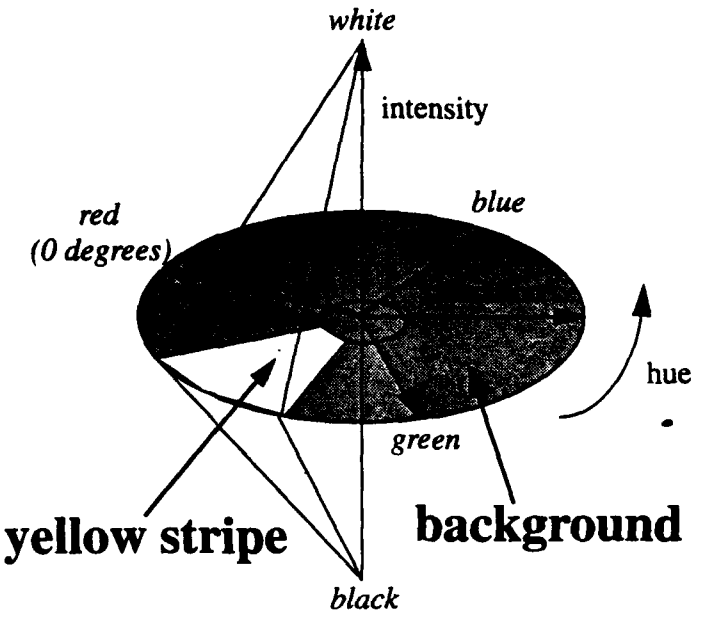



**Figure 12: Classification of pixels as yellow stripe or background based on hue and saturation**

The planes in (*R, G, B*) space which correspond to the minimum and maximum hue values for yellow stripe pixels are computed the first time the tracker is run. Pixel classification is done by plugging the pixel (*R, G, B*) value into the two plane equations. If $((R-B)+(B-G)\times R_{minhue}) < 0$ and $((R-B)+(B-G)\times R_{maxhue}) > 0$ (the pixel falls in the specified hue range) and $1-(3\times min(R,G,B)/(R+G+B)) > minsat$ (the pixel's color is not too close to grey) then the pixel is classified as yellow stripe. Otherwise the pixel is classified as background. Doing the classification this way rather than by explicitly computing the hue of the pixel eliminates the need to call the arccos function for every pixel, and results in rapid classification.

### 3.3.2 Tracking white painted stripes

While color provides a fairly strong invariant characteristic of painted yellow stripes, it provides a very poor cue for distinguishing between grey pavement and a white stripe. Intensity contrast is a much more reliable cue for detecting the location of white stripes. Two trackers have been created and tested for the robust detection of white painted stripes. Both average the image in the prediction window along the expected direction of the feature in order to improve the clarity of the position of the stripe in the image signal. The first convolves the blurred signal with a filter which looks for a bright bar of a specified width (the *oriented bar* tracker). The second looks for step edges in the blurred signal which have opposite sign, similar magnitude, and whose separation is within some tolerance of the expected feature width (the *matched edge* tracker).

### 3.3.2.1 The oriented bar tracker

The blue band of the image is averaged parallel to the expected orientation of the stripe in order to reduce variation in the signal caused by shadows, stains, puddles, and other extraneous distracting features. This averaging enhances the contrast of the stripe with the background and reduces the 2-D search window to a 1-D signal. A parallelogram shaped window is used rather than an oriented rectangular window in order to make the averaging operation fast and avoid the need for precomputed masks. Rows are fetched offset by the expected slope of the stripe for orientations within 45 degrees of vertical in the image. Corresponding columns of the blue band are averaged to produce a one dimensional signal. In the case of a predicted feature orientation greater than 45 degrees from vertical in the image, columns are fetched and the corresponding rows averaged to produce the one dimensional signal. The white stripe appears in this one dimensional signal as a plateau bordered by sharp step edges.

The location of the plateau is detected by cross correlation with a center-surround filter. The filter has a central region with weights of +1 bordered on either side by regions with weights of -1. The width of the central (positive weight) region matches the expected width of the stripe. The negative weight region on either side has half the width of the expected stripe. Using weights of +1 and -1 allows the speeding up of the cross correlation through the use of partial sums. Figure 13 graphically illustrates the change in the cross correlation sum which occurs as the mask moves one pixel to the right. The leftmost pixel drops out of the window, and is added back into the sum. The new rightmost pixel is subtracted from the sum. The pixels which change sign as the borders between positive and negative weights shift are added or subtracted from the sum appropriately. The cross correlation value is not normalized.

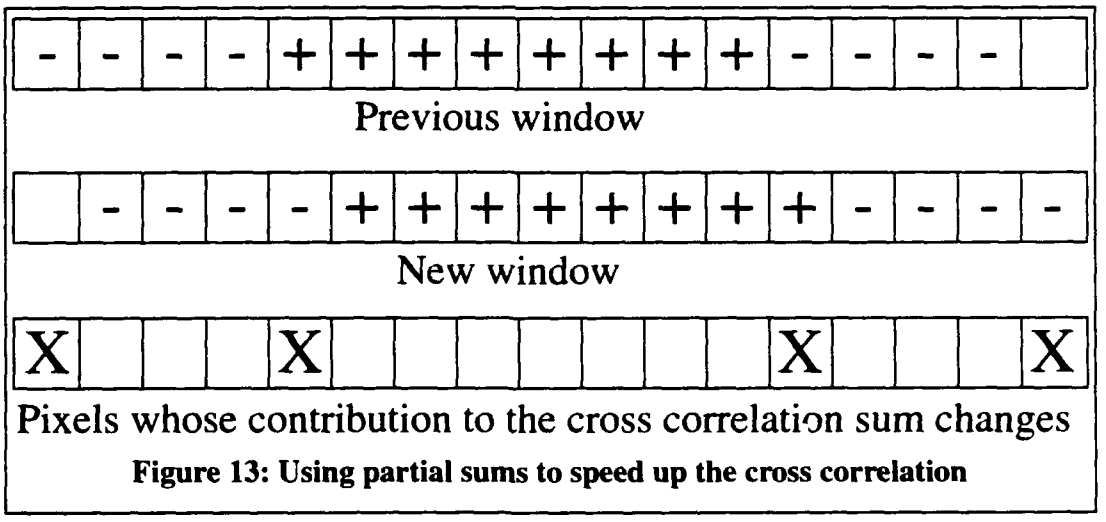



**Figure 13: Using partial sums to speed up the cross correlation**

The maximum and minimum values of the cross correlation are located. Two criteria must be met for the maximum to be considered valid. The first is that the maximum must exceed a specified threshold. The second is that the maximum must exceed the absolute value of the minimum. If the maximum passes both of these tests, then the location of the maximum is reported as the feature location. If either the absolute or relative value of the cross correlation peak is too low, then the tracker returns an error value indicating that the stripe was missing in the window. Figure 14 summarizes the processing steps in the oriented bar tracker.

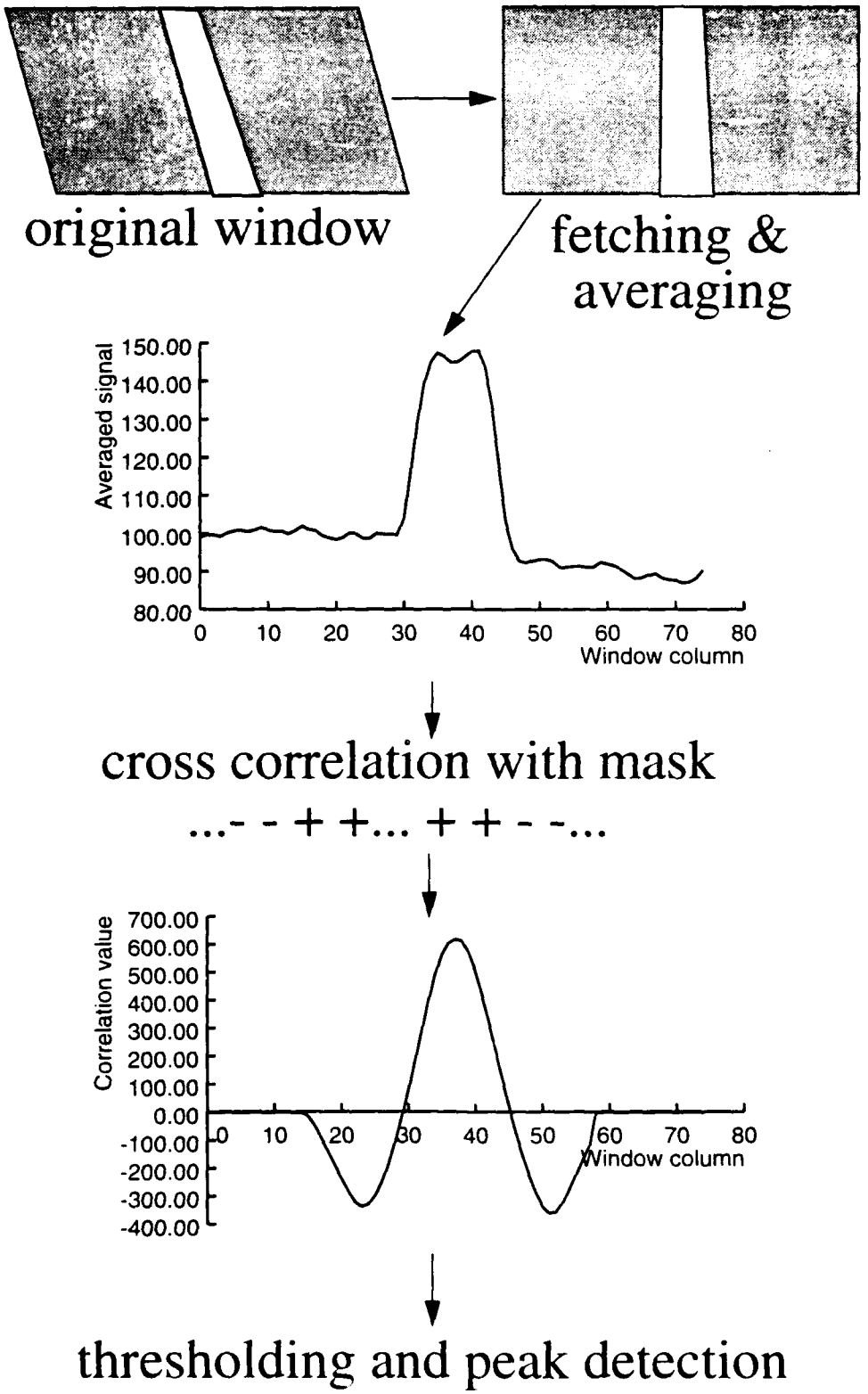



**Figure 14: Description of oriented bar tracker algorithm**

Errors in the predicted stripe width and orientation affect the magnitude of the maximum cross correlation value. Aubert [1] describes experiments comparing the peak cross correlation values for operators of various widths at both the correct orientation and at an orientation in error by 10 degrees. Such prediction errors can produce false negative responses from the tracker by reducing the value of the cross correlation peak below the tracker's acceptance threshold. His conclusion was that the tracker could tolerate an error in orientation estimate of approximately 10 degrees and that it is desirable to inflate the width estimate slightly to reduce sensitivity to underestimating the stripe width.

False positives can occur if the window contains a step edge of the proper orientation. If the stripe is actually present in the window and the distracting edge has sufficiently higher contrast, then the wrong location will be reported for the feature. If the stripe is absent then two problems occur. The first is the addition of a contaminant into the feature position data. The second is the failure to recognize that the feature was absent at the predicted location. The next tracker was created in order to avoid such false positives.

### 3.3.2.2 The matched edge tracker

The oriented bar tracker is unable to distinguish between peaks in the cross correlation function which correspond to the target shape in the signal (a plateau of specified width bounded by step edges) and other signal shapes which produce above threshold responses. The matched edge tracker was created to eliminate such false positives. It looks for the white stripe by running a simple one-dimensional edge detector over the averaged signal and looking for a pair of edges with opposite contrast separated by the expected stripe width.

The algorithm for the matched edge tracker is very similar to the algorithm for the oriented bar tracker. The same staggered fetching and averaging is done to produce a one-dimensional blue signal. Instead of the center-surround mask used in the oriented bar tracker, a mask with a left negative half and right positive half is used. The width of the mask is equal to the expected width of the stripe. The weights used are +1 and -1 as in the oriented bar tracker to allow fast cross correlation. The cross correlation value is normalized by dividing the raw value by half the width of the mask, producing a value which represents the blue intensity change.

The algorithm then locates extrema in the output of the cross correlation with the edge detector filter. This results in a list of candidate edges in the signal. This set of edges is searched for a pair of edges which meets the following three criteria:

- the edges have opposite contrast, with the signal brighter between the two edges,
- the edges are separated by a distance within a specified percentage of the expected feature width, and
- the absolute magnitudes of the two edges are similar (specifically, let $M_P$ be the magnitude of the positive edge and let $M_N$ be the absolute value of the magnitude of the negative edge -- the pair is accepted if $|M_P - M_N| < \alpha \times (M_P + M_N)$ ).

The tracker indicates that the feature was absent if no pair of edges in the window meets these criteria. Figure 15 shows the stages of processing in the algorithm for this tracker.

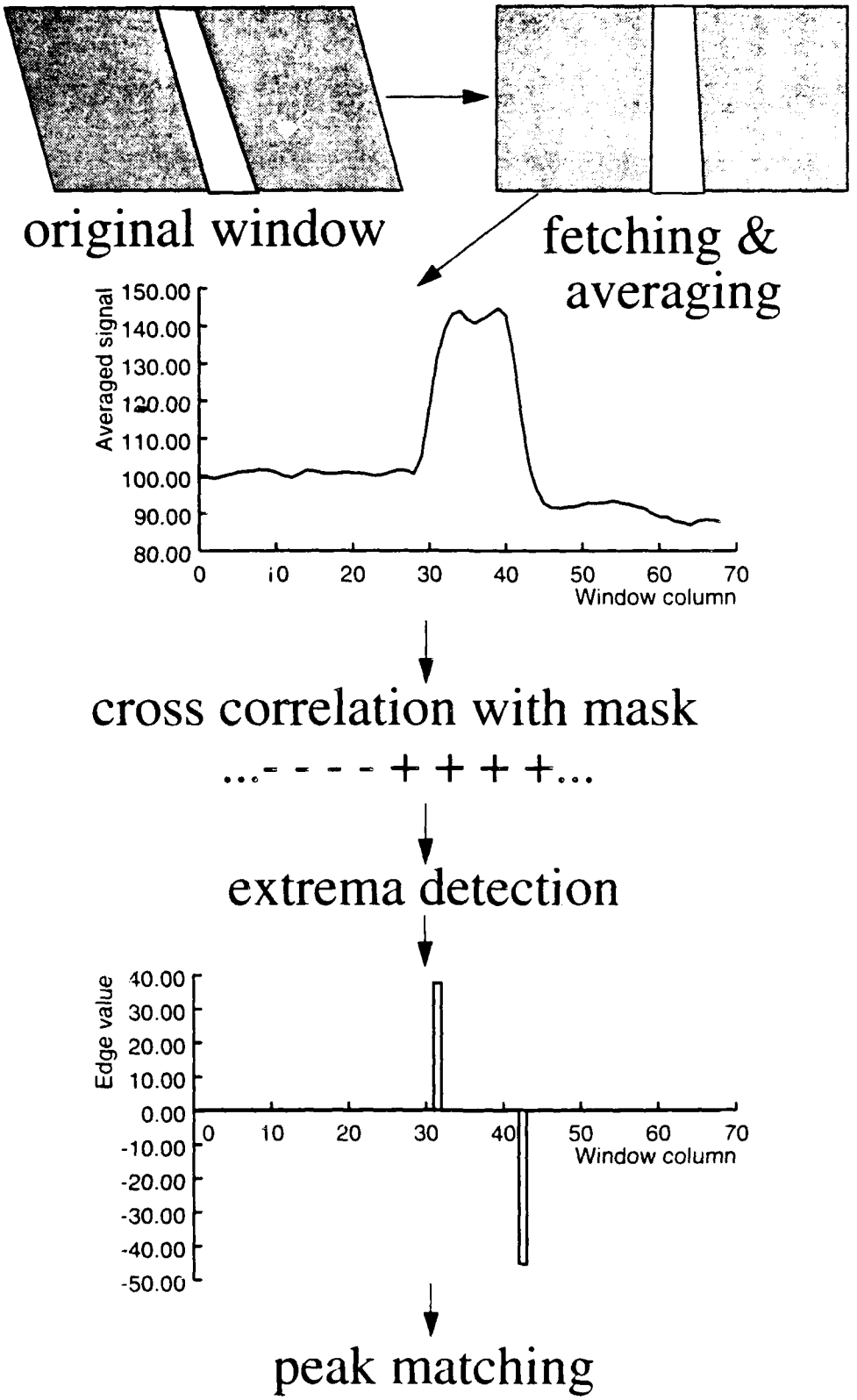



**Figure 15: Processing for the matched edge tracker**

The matched edge tracker shows a slightly higher rate of false negative responses than the oriented bar tracker. This is due to the dependence on similar edge magnitudes as well as expected width. If the pavement is brighter on one side of the stripe than the other the edges may not have sufficiently similar magnitudes. The rate of false positive responses is much smaller than that of the oriented bar tracker, however.

Figure 16 illustrates a situation in which the oriented bar tracker generates a false positive response, while tne matched edge tracker correctly determines that the white stripe is absent. The white stripe was obscured by snow in the specified search window, and was not visible. The middle plot in Figure 16 shows the value of the cross correlation function for the oriented bar tracker. This tracker responds strongly to the broad intensity peak between columns 0 and 20, and less strongly to the feature between columns 40 and 55. The maximum cross correlation value is well above threshold even though the intensity function looks nothing like the sharp plateau the filter is intended to locate.

The matched edge filter is able to detect the absence of the white stripe because it does not find a matching pair of edges of similar contrast and opposite sign. The lower plot in Figure 16 shows the location and magnitude of the local maxima in the edge filter response in the matched edge tracker. In this window there are no locations where the edge filter returned a positive value, as the signal is monotonically decreasing at the scale at which it was smoothed by the edge detector.

### 3.3.3 Tracking ragged road/shoulder boundaries

The edge between the pavement surface and the adjacent terrain is often ragged. In addition, the off-road area is often highly textured (gravel or grass shoulders, for example). As a result, edge detectors which look at a small window are inappropriate for detecting such boundaries. Instead, the same simple oriented edge detector used in the matched edge tracker is used to find such boundaries. The averaging parallel to the expected feature orientation reduces the effects of texture and compensates for deviation of the boundary from a straight edge.

The width of the edge mask is selected so that the smoothing of the filter is done at a constant spatial frequency in the ground plane. This is done by dividing a fixed width by the width of a pixel on the ground plane along the center row of the search window to get the width of the mask in pixels for that window. The mask is cross correlated with the averaged blue signal just as in the matched edge tracker. If the largest edge magnitude in the window is above threshold, that position is returned as the feature location.

This is unquestionably the least reliable of the trackers described. Noise edges can have higher contrast than the road boundary, making it impossible to select a threshold to avoid false positives without generating false negatives. In general, tracking pavement/terrain boundaries with local window operators appears to be very difficult. Only the VaMoRs system claims to be able to robustly follow pavement edges using a local window operator. It is difficult to assess to what extent their success is due to the tracker (a simple oriented edge tracker) and to what extent it is due to the care they put into the formulation of their data filtering, which includes modeling the rate of change of the road curvature.

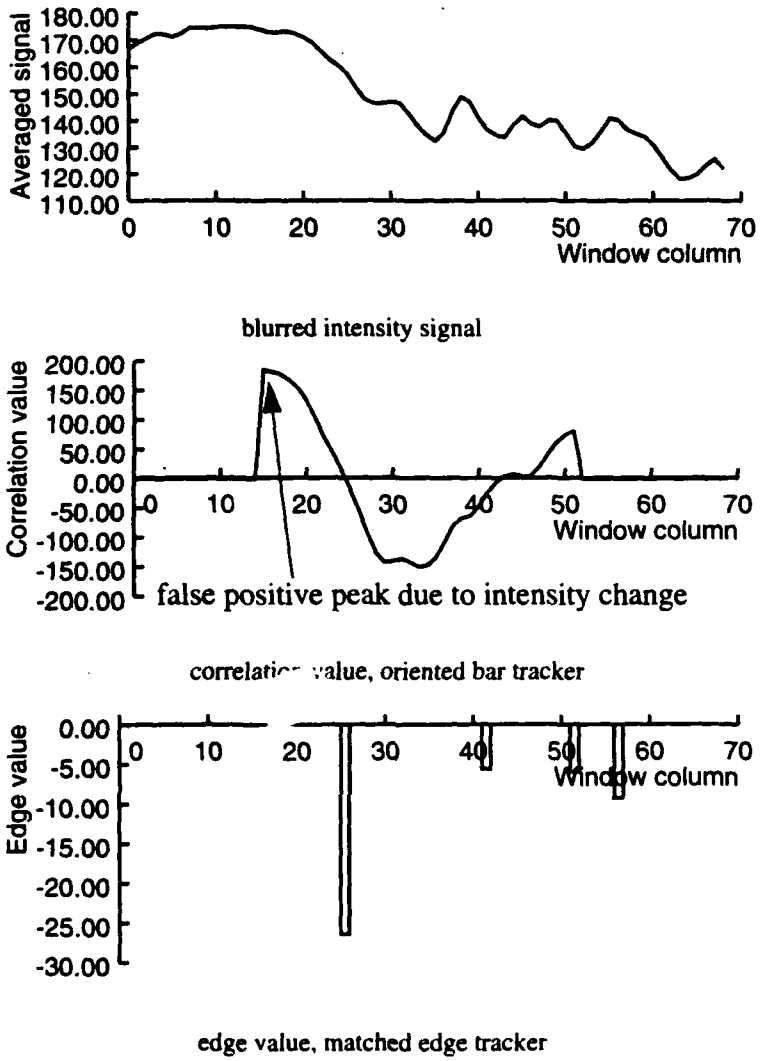



**Figure 16: Comparison of oriented bar and matched edges trackers on window where the white stripe is not visible**

## 3.4 Performance evaluation

Tracker performance has a number of dimensions. The most important, but most difficult to evaluate, is the robustness of an operator under varying environmental conditions. Next is the locational accuracy of an operator. Discrepancies between the system's models of various aspects of the world and the world itself, as well as errors due to the segmentation techniques used, create errors in the estimates of feature locations returned by the trackers. The final dimension of performance considered here is tracker execution time. In the presence of limited computational resources it may be necessary to weigh the utility of examining a given area of the image against the cost in processing time, requiring the ability to model the execution time of the various segmentation techniques used. YARF is instrumented to aid in the evaluation of all three of these dimensions of performance.

### 3.4.1 Estimation of environmental robustness of trackers

Heuristic segmentation techniques are not based on analytic theories of how scene and illuminant properties interact with the sensor to produce the image, making it impossible to specify the limits of their performance analytically. As a result, the only way to evaluate this dimension of performance is by performing experimental runs under a variety of environmental conditions and seeing how the trackers perform.

YARF provides routines which can be invoked off-line to examine tracker performance on test images. As an example, histograms of hue and saturation in an image window can be produced to examine cases where the usual thresholds appear to be incorrect. Profiles of the averaged blue signal in an oriented window and cross correlation values with the operator mask can be produced for the oriented bar, matched edge, and oriented step edge trackers (this is the source of the profiles in Figure 14 and Figure 15) in order to analyze cases where they give incorrect results.

Future work in this area would involve a systematic testing of the trackers. Data would be gathered at a set time daily over a period of a year or so in an effort to see as broad a range of weather situations as possible. One possible barrier to such a plan would be the commitment of researcher and vehicle time involved in such experimentation. Opportunistic experimentation taking advantage of bad weather is only a partial answer, as vehicles are shared resources whose time needs to be scheduled in advance for use by other experimenters or for maintenance. High quality recorded video data may be able to solve this problem, as the goal here is purely to test the trackers, and not other aspects of the system such as the model fitting or path tracking.

### 3.4.2 Estimation of tracker feature localization error

Estimation of the variance of the errors in feature position estimation for the different tracker types is more difficult. Ground truth is not known. The system processes a large number of images (on the order of 500 for every kilometer travelled), making comparison with ground truth difficult even if ground truth were known. Examination of data from Navlab runs showing the backprojection onto the assumed ground plane of tracker data from multiple frames indicates that errors in calibration, feature position localization, and estimation of vehicle motion between frames do not appear to be significant. An example is shown in Figure 17. Data from eight images have been backprojected onto the ground plane assumed by the calibration and placed into a common coordinate system using the estimate of vehicle motion between frames supplied by the Navlab controller. The squares in the figure are individual feature positions returned by the trackers. Note how well they line up as the vehicle traverses the gentle curve in the road.

While experiments have not been done to measure the error in feature localization relative to ground truth in test images, a statistical measure of the coherence of the data as illustrated in Figure 17 has been implemented. The system computes and periodically prints the variance of the distance between the individual tracker data points projected onto the ground plane and the estimated location of the corresponding road features. This provides a measure of how much of the individual feature point locations are accounted for by the stripe model. In a typical run with the Navlab II, this produced a standard deviation of 4.37 inches for points from the yellow hue tracker and a standard deviation of 3.93 inches for points from the oriented bar tracker. In general, the standard deviation is usually under 6 inches, consistent

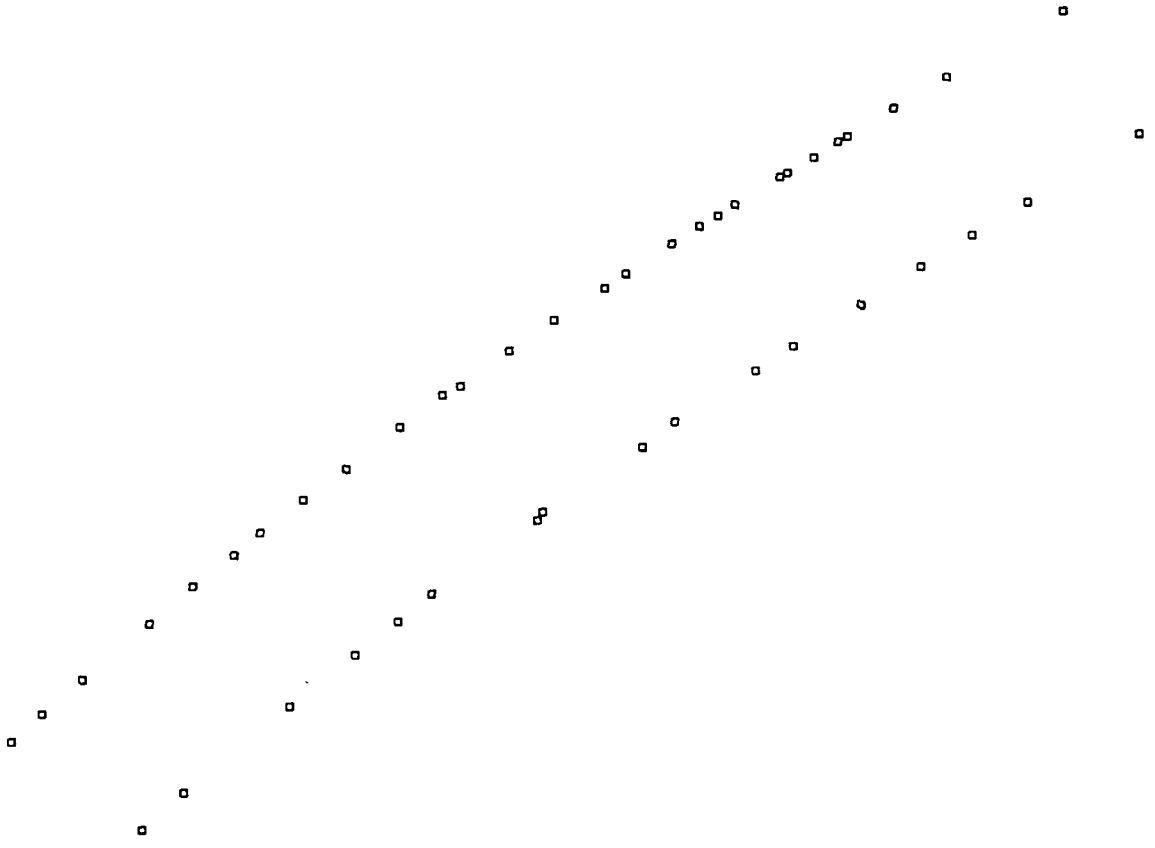


**Figure 17: Tracker results integrated over multiple frames as the vehicle moves**

with the coherence illustrated visually in Figure 17. These standard deviations reflect all of the sources of error in the system, including errors in feature localization in the image plane, errors introduced by the flat ground plane model, errors in map updating due to noise in the controller's estimate of vehicle motion, and errors due to small changes in lane width. For comparison, the errors in human drivers' estimates of their lateral offset from the lane center typically have standard deviations of 6.54 to 14.3 inches [5].

### 3.4.3 Estimation of tracker execution time

Timing routines are provided to fit a simple model of tracker execution time to timing data computed for each window processed by a tracker. The model of window execution time is $time = pixel\_cost \times area + overhead$, or $time = \begin{bmatrix} area & 1 \end{bmatrix} \times \begin{bmatrix} pixel\_cost & overhead \end{bmatrix}^T$ in matrix form. Each tracker filters (execution time in seconds, window area in pixels) pairs to update the estimate of per-pixel cost and per-window overhead for that tracker. Each tracker also records the number of pixels examined with that tracker in each frame.

The timing estimation filter periodically prints the current estimates of model parameters for each tracker being used in a run. The following example of the output was taken from a YARF run on the Navlab II vehicle. YARF was running on a Sparc 2 with 32 megabytes of memory, using the yellow hue and oriented bar operators. After 205 frames the parameter

estimate for the yellow hue operator was $[pixel\_cost\ overhead]^T = [2.59\times10^{-5}\ 8.42\times10^{-4}]^T$, with covariance $\begin{bmatrix} 1.25\times10^{-12} & -1.61\times10^{-9} \\ -1.61\times10^{-9} & 2.17\times10^{-6} \end{bmatrix}$. The parameter estimate for the oriented bar operator was $[pixel\_cost\ overhead]^T = [1.03\times10^{-5}\ 1.68\times10^{-3}]^T$, with covariance $\begin{bmatrix} 1.21\times10^{-13} & -6.05\times10^{-10} \\ -6.05\times10^{-10} & 3.16\times10^{-6} \end{bmatrix}$.

While YARF does not currently use this information, it is available for later use in extensions of this research. YARF does not take processor speed into account and drives at a constant speed set by the user at the beginning of the run. Such models of tracker execution time could be used in conjunction with an estimate of the utility of checking a particular image window to select which windows to examine given limited processing time, and to slow the vehicle if the time needed to verify essential features exceeds the current image cycle time.

## 3.5 The advantage of using multiple trackers

One of the major claims of this thesis is that performance can be improved by using model information about feature type and appearance to guide the application of specialized feature trackers. The following experiment was performed in order to demonstrate the added effectiveness of using multiple specialized feature trackers.

Sixteen images were examined from a sequence taken on a divided road. The double yellow line in the center and the white stripe on the right side of the lane were used as test features. The position of each feature was picked out by hand using a cursor, and trackers placed at preselected rows along the feature. The rate of positive responses for each tracker was computed from the data. The results are summarized in the table shown in Figure 18.

| Tracker type | % positive response to double yellow stripe | % positive response to white stripe |
|---|---|---|
| yellow hue tracker | 94 | ** |
| oriented bar tracker | 28 | 91 |
| matched edge tracker | 37 | 84 |
| oriented bar, 43% usual thresh. | 72 | ** |
| matched bar, 50% usual thresh. | 62 | ** |

**Figure 18: Relative tracker reliability**

The trackers compared for the double yellow stripe were the yellow hue operator; the oriented bar and matched edge operators with the thresholds normally used for white stripes; and the oriented bar and matched edge operators with reduced thresholds to compensate for the lower contrast of yellow stripes. The yellow hue operator detected the double yellow line in 94% of the search windows. The contrast based operators performed much worse.

With the peak magnitude threshold used to track white stripes, the oriented bar operator detected the double yellow line in 28% of the search windows. Reducing the threshold to 43% of the usual value increased the detection rate to 72%. With the edge magnitude threshold used to track white stripes, the matched edge operator detected the double yellow line in 37% of the search windows. Reducing the threshold to 50% of the usual value increased the detection rate to 62%. Thus, by having a specialized feature tracker to detect yellow stripes, YARF is able to more reliably detect this type of feature than if it had only one or both of the contrast based operators.

The trackers compared for the white stripe were the oriented bar and matched edge operators. Both perform well in detecting the white stripe. The oriented bar operator detected it in 91% of the search windows, while the matched edge operator detected it in 84% of the search windows. As was mentioned above, this slightly lower success rate for the matched edge tracker is due largely to the contrast match constraint.

## 3.6 Conclusion

The YARF approach to detecting road features in images takes advantage of constraints provided by the system's models of road geometry and appearance to simplify the segmentation problem. Expected feature locations and orientations provided by the model allow more reliable detection of white painted stripes and pavement boundaries by restricting consideration to edges with the proper orientation near the expected location. Model knowledge of feature types permits the use of simple, specialized segmentation techniques exploiting simple models of the appearance different feature types. Yellow stripe detection uses hue invariance as a cue, while white stripe detection uses expected width and orientation in conjunction with image contrast.

The specialization of trackers to detect particular feature types, and the ability of the trackers to indicate that the feature appeared to be absent in the given search window combine with model information about feature types to provide a basis for discriminating between features in the way needed to support the semantics of the road following task. The ability to detect the absence of particular types of features also provides support for the capability to detect changes in lane structure and the approach of intersections, as will be described in a later chapter.

# 4 Road model parameter estimation

The trackers described in the previous chapter return image coordinates which correspond to locations on the features defining the road. Those individual measurements must then be combined into a unified estimate of the location of the vehicle on the road and the local road geometry parameters (in this case, the spine curvature). The constraints provided by the generalized stripe model of the road geometry generate the system of equations relating individual feature point measurements to the vehicle position and road curvature.

The first section of this chapter presents a review of the different techniques other vision-based autonomous road following systems use to extract the scene geometry from image segmentation data. These fall broadly into four categories: backprojection, differential, Hough, and statistical methods. Backprojection and differential methods tend to be sensitive to noise in the segmentation results. Hough techniques are difficult to apply to models with more than two parameters, and peak detection is sometimes difficult. Statistical methods appear to be superior. They provide a well defined criterion for selecting model parameters given noisy data, and have well understood properties.

The second section describes the integration of feature location data from multiple image frames into a local map. This local map serves several roles in YARF. It is used to combine data from multiple images for road model parameter estimation. It is also used to record information about locations where expected features were not detected for analysis by the map navigation module of the system to detect changes in lane structure and intersections.

The third section analyzes the errors introduced by approximations to the model for parameter estimation. While the nonlinear equations defining circular arcs are used to predict feature locations for the trackers, for estimation purposes approximations are made to the equations in order to have a system which is linear in its parameters. Several other road following systems use similar linearized models. This introduces errors which affect the accuracy of the recovered road location and shape. Experiments demonstrate that the resulting errors can be significantly reduced by performing the estimation in a data-dependent coordinate system. This process, which rotates the data into a coordinate system chosen to minimize the error, is referred to as *virtual panning*.

The final section describes the use of robust estimation to handle situations where contaminants occur in the data which would otherwise cause the system to lose tracking of the road. The data provided by the feature trackers is not perfect. In order to extend the reliability of road tracking it is necessary to use an estimation technique which is more robust than standard least squares techniques in situations where there are contaminating false data observations. YARF demonstrates the use of Least Median of Squares estimation for road following, and the ability of LMS estimation to avoid errors which would cause a system using standard least squares estimation to fail.

## 4.1 Methods for recovering model parameters

Four main types of methods have been used to recover road model parameters from image segmentation data: boundary backprojection, differential reconstruction, voting in the model parameter space, and statistical fitting techniques.

In boundary backprojection methods, image features detected by the segmentation are backprojected onto the (assumed) ground plane, and consistency constraints are applied to determine which features are part of the road. This is the method used in the VITS [54], FMC [31], and U. Bristol [44] systems. These techniques are intrinsically limited by the flat earth assumption, which restricts their ability to successfully extract the road in scenes with significant changes in road elevation.

Differential reconstruction algorithms recover three dimensional road structure using assumptions of constant road width and zero road bank [11][25]. These algorithms do not enforce higher level constraints on relative feature location. As a result, errors in the image segmentation are interpreted as changes in road slope due to the constant width assumption. This can produce large errors in the recovered road shape. Algorithms which sample the road edges create a tiling problem, in which poorly chosen samples result in a road reconstruction which smooths over terrain features. The results of a comparison of flat-earth reconstruction with two differential methods are presented in [36].

In parameter space voting techniques, detected feature locations vote for all possible roads they are consistent with. This method is used in SCARF [10], in algorithms developed at the University of Michigan [33][39], and in some of the LANELOK [27][28] algorithms. The main advantage of these techniques is their robustness in the face of large amounts of noise in the segmentation results. The main disadvantage is the difficulty of using voting for models which have more than two or three parameters, resulting in large multidimensional Hough spaces. Also, peak detection in the accumulator space can be difficult. Noise in the data can produce structure in the accumulator array which prevents the use of simple blurring to enhance the peak.

Neural network techniques can be though of as a form of parameter space voting technique. The ALVINN system [40] is a neural net architecture for road following which learns to drive on different types of roads by providing a reduced resolution image as input to the net and training the system to emulate the steering behavior of a human driver. The output of the network represents the steering direction which the vehicle should use for the camera input shown. Thus, the network is parameterizing road scenes by the steering direction which causes the vehicle to track the road. The system appears capable of extracting appropriate features on a broad variety of road types, and has successfully driven the Navlab II vehicle for a distance of 21.2 miles on an unmodified public highway near Pittsburgh. ALVINN has no model of road shape or the location of the vehicle on the road. As a result, heuristics have to be used to integrate ALVINN into symbolic systems which require such information. Also, it is an open question whether there is a reasonably small closed set of networks which can handle general road navigation on a large set of different roads.

In statistical fitting procedures, road model parameters are fit using the observed data points and the equations of the road model. While some systems which use statistical fitting techniques backproject the feature data onto a flat ground plane, they differ from systems which use boundary backprojection by imposing the stronger requirement that the road spine belongs to a specified family of parametric curves. Standard techniques such as least squares or robust techniques which are less sensitive to outlying data observations can be used. VaMoRs [13], YARF, and other of the LANELOK algorithms use this type of technique. Statistical fitting techniques have a number of advantages over the other available techniques for model parameter recovery. They enforce smoothness in the presence of noise in the data. They are computationally efficient and they have a vast literature of theory, techniques, and tools associated with them.

## 4.2 Constructing the local map

YARF integrates feature data from multiple frames by constructing a map of the local environment. This map is used rather than a filter based approach for two reasons. First, existing robust estimation techniques are batch methods. Second, the map is needed by other modules in YARF for other purposes.

This map represents the feature points in a vehicle-centered coordinate system. As data points are detected in new images they are entered into the map, and as data points from old images move outside the boundary of the map they are deleted. There are two steps involved in maintaining the local map of feature data. The first is the transformation of the data from previous images in the vehicle-based coordinate system using the estimate of vehicle motion provided by the controller. The second is the projection of the image coordinates of feature points in the current image onto the local ground plane in order to add them into the map.

The vehicle controllers on the Navlab I and Navlab II vehicles maintain an estimate of vehicle position and orientation. The difference in estimated vehicle position and orientation between the digitization of the current image and the digitization of the previous image is used to update the locations of points from previous images in the vehicle-based coordinate system used in the local map. The origin of this coordinate system is the $(x, y)$ location of the focal point of the camera on the ground plane.

The process of constructing the local map is illustrated in Figure 19. The three diagrams show the local map for three successive images during a run. The picture on the left (rotating the page so that the captions are upright) shows data from a single frame backprojected onto the ground plane. The rectangle at the bottom is the Navlab. The trapezoid at the top is the field of view of the camera cropped at the farthest distance any of the features is being tracked in the image. The line along the left edge has markings every two meters to provide scale. The small diamonds are feature locations returned by the trackers. The first digit of the number next to each feature point is the frame number mod 10, the second digit is the number of the feature in the road model. The middle picture shows the map for the next frame, frame seven. The points from frame six have been transformed based on the vehicle controller's estimate of the motion between the frames, and the new data points have been backprojected onto the ground plane and added into the map. The right picture shows the process continuing for the next frame. Data points are dropped from the map when they pass a specified distance behind the vehicle.

The coverage of each feature becomes fairly dense as additional frames are processed. Several instances can be seen in Figure 19 where windows from different frames fell close to each other on the ground plane. In these cases the feature locations are very close to each other, giving an indication of the accuracy of the camera calibration and vehicle position estimation.

The camera model used in YARF assumes a flat ground plane and perspective projection. The camera is tilted with respect to the horizon, but there is no roll. Figure 20 illustrates the geometry relating a pixel in the image plane to a point on the ground plane. The image plane coordinate system is defined with its origin $(0, 0)$ in the upper left corner of the image. Row numbers increase towards the bottom of the image, column numbers increase towards the

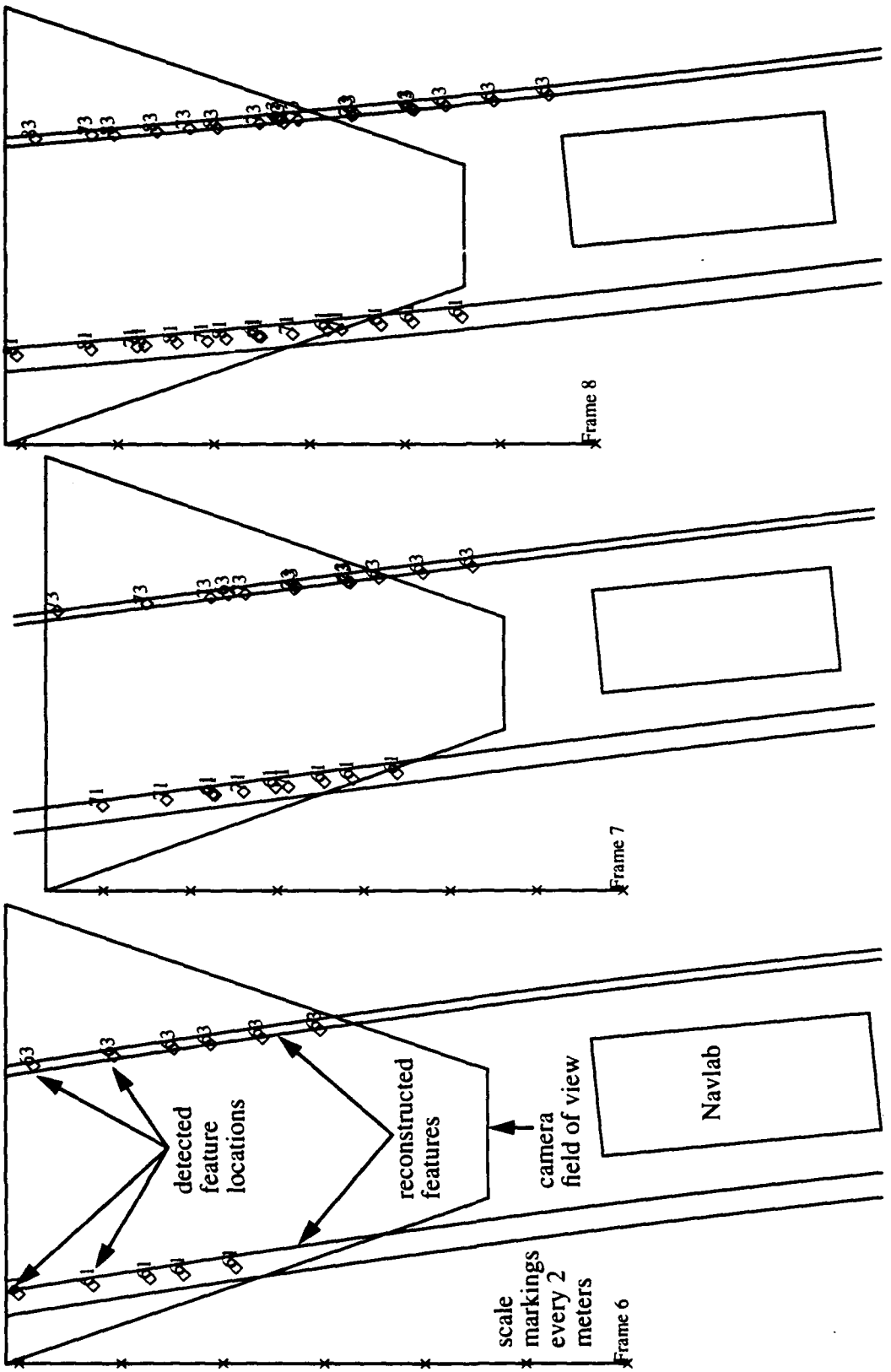



**Figure 19: Integration of data from multiple images into a local map**

right side of the image. The camera ground frame in the ground plane has its origin directly below the focal point of the camera. The projection onto the ground plane of a ray from the focal point through the center of the image defines the *y* axis of the camera ground frame.

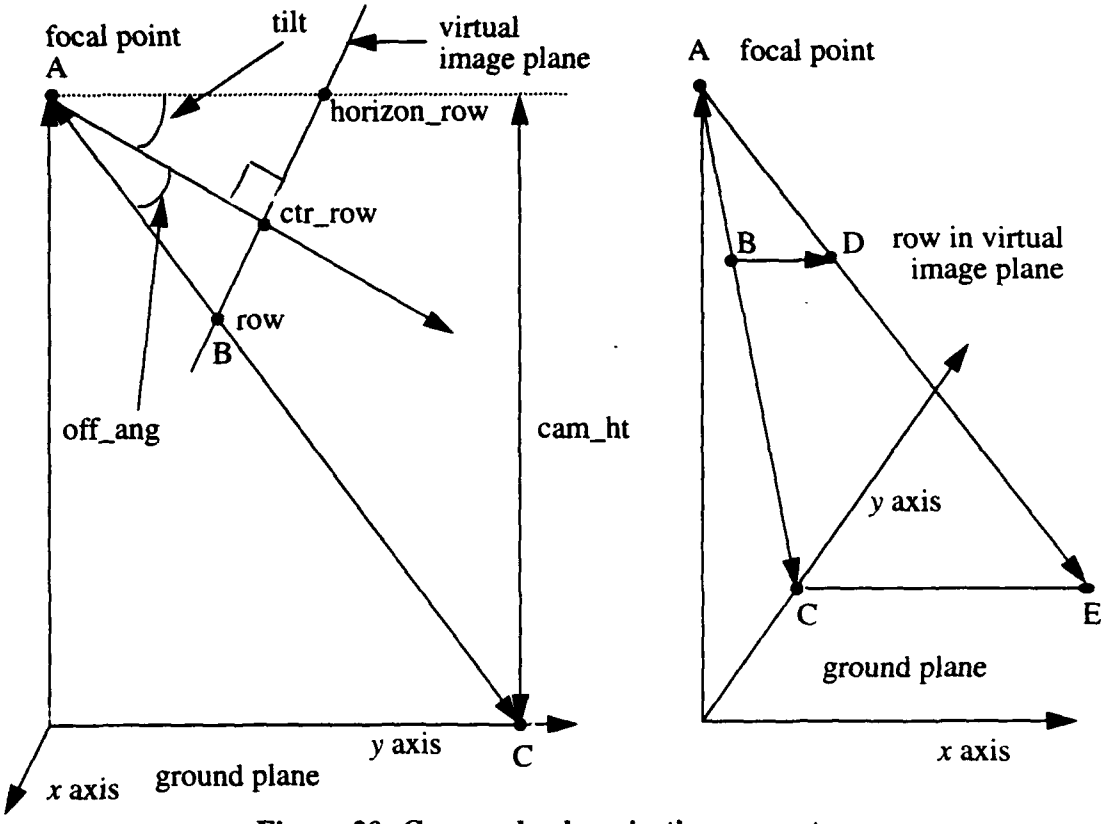



**Figure 20: Camera backprojection geometry**

The first step in converting between image coordinates and ground coordinates is the conversion between image row and ground y. Let $rf = pixel\_height / focal\_length$. The tangents of *off_ang* and *tilt* are given by $\tan(off\_ang) = rf \times (row - ctr\_row)$ and $\tan(tilt) = rf \times (ctr\_row - horizon\_row)$. By the definition of the tangent of an angle in a right triangle, $y = cam\_ht / (\tan(tilt + off\_ang))$. Applying the trigonometric identity for the tangent of the sum of two angles yields the formula

$$y = \frac{cam\_ht \times (1 - rf^2 \times (row - ctr\_row) \times (ctr\_row - horizon\_row))}{rf \times (row - horizon\_row)}.$$

The formula for converting from *y* in the camera ground frame to image row can be obtained by some simple algebraic manipulation of the above formula. Define $k1 = rf^2 \times cam\_ht \times (ctr\_row - horizon\_row)$, $k2 = cam\_ht + k1 \times ctr\_row$, and $k3 = rf \times horizon\_row$. Then the image row corresponding to a given *y* value in the camera-based ground frame is: $row = \frac{y \times k3 + k2}{y \times rf + k1}$.

The relationship between the image column of a point and the $x$ value of the corresponding point on the ground plane is defined by the similar triangles ABD and ACE. Since the ratios of the lengths of corresponding sides are equal, we know that $(AC/CE) = (AB/BD)$. The length of side AC, the ray from the focal point to the ground plane through the center column of the row on which the pixel lies, is $\sqrt{y^2 + cam\_ht^2}$. The length of AB, the ray from the focal point to the center of the row of the pixel in the image plane, is $\sqrt{focal\_length^2 + (row - ctr\_row)^2 \times pixel\_height^2}$. The length of BD, the distance from the center column of the row to the column of the image pixel, is given by $pixel\_width \times (column - ctr\_col)$. Substituting these lengths into the equation relating the lengths of the sides and solving for the length of CE, the $x$ coordinate of the point on the ground plane corresponding to the pixel, yields the equation:

$$CE = x = \frac{(column - ctr\_col) \times pixel\_width \times \sqrt{y^2 + cam\_ht^2}}{focal\_length \times \sqrt{1 + rf^2 \times (row - ctr\_row)^2}}$$

Defining $cf = pixel\_width / focal\_length$ this becomes:

$$x = cf \times (column - ctr\_col) \times \sqrt{\frac{y^2 + cam\_ht^2}{1 + rf^2 \times (row - ctr\_row)^2}}$$

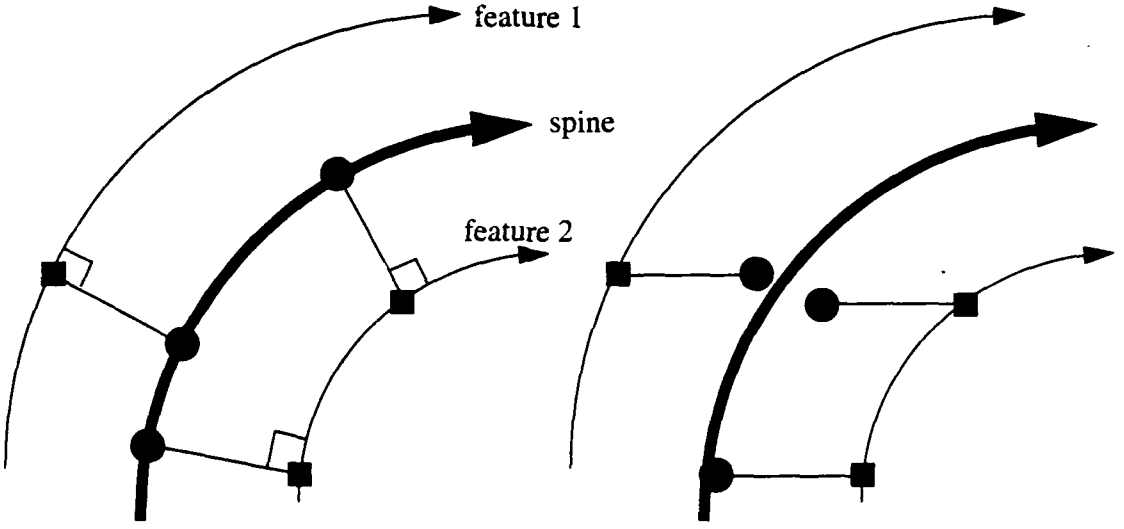
Calibrating the camera using this model involves measuring the camera height, the horizon row of the image, the center row and column values, and the row and column pixel size factors *rf* and *cf*. The camera height can be measured directly with a measuring tape. The horizon row is estimated by looking at a straight stretch of road, selecting points on the left and right edge of the road, and computing the row of the vanishing point. The model assumes that the principle ray passes through the center of the image, so the center row and column values are known given the choice of image coordinate system. The convention used places *(0, 0)* in the upper left corner. The row number increases moving downward in the image, and the column number increases moving to the right. Measuring the $y$ value corresponding to the center row of the image yields the *rf* value. Measuring the number of columns corresponding to a known width on the ground yields the *cf* value. The translation and rotation between the camera-based frame on the ground plane and the coordinate system of the vehicle (with its origin at the center of the rear axle) are measured separately.

## 4.3 Road model and parameter fitting used in YARF

Section 2.4 described in detail the road model used in YARF. A one-dimensional feature cross section is swept perpendicular to a circular arc spine curve in the ground plane. Figure 21 shows an example of detected feature locations in an image, and the road geometry recovered from those feature locations using the constraints provided by the stripe road

model. For estimation purposes two approximations are made to the equations defining the road model in order to have a systems of equations which is linear in the parameters being estimated.

The first approximation simplifies the form of the spine arc. The circular arc spine curve is approximated by a parabola. A parabola represents the first three terms of the binomial series expansion of the equation of a circular arc. The second approximation involves the relationship between individual feature locations and the spine curve. YARF estimates the parameters of the spine arc of the road rather than fitting each feature separately. This requires relating the points on the different features being tracked to the location of the spine curve. The point on the spine which corresponds to a given point on a feature lies along a line passing through the feature point and perpendicular to the feature tangent at that point. Rather than use this nonlinear relationship and an iterative fitting procedure, YARF translates the data points parallel to the $x$ axis to bring them approximately onto the spine arc. This is illustrated in Figure 22 below.




feature points corrected to lie on the spine by translating them perpendicular to the feature tangent at each point

approximation used in YARF, with feature points corrected to lie on the spine by translating them parallel to the $x$ axis

**Figure 22: Feature to spine correction approximation in YARF**

The model that results from these approximations is $x \simeq 0.5 \times curvature \times y^2 + head\_tan \times y + spinetrans - offset$. $(x, y)$ is the position on the ground plane of a detected feature point. *Offset* is the offset of the feature from the road spine. *Curvature* is the curvature of the spine arc. *Head_tan* is related to *heading*, the tangent of the spine arc at the x-intercept, by the formula $heading = \text{atan}(1/\text{head\_tan}) - (\pi/2)$. This corrects for the rotation due to the estimation being performed with $x$ as a function of $y$. *Spinetrans* is the $x$-intercept of the spine arc. Given a set of data points on the ground plane which lie on different features in

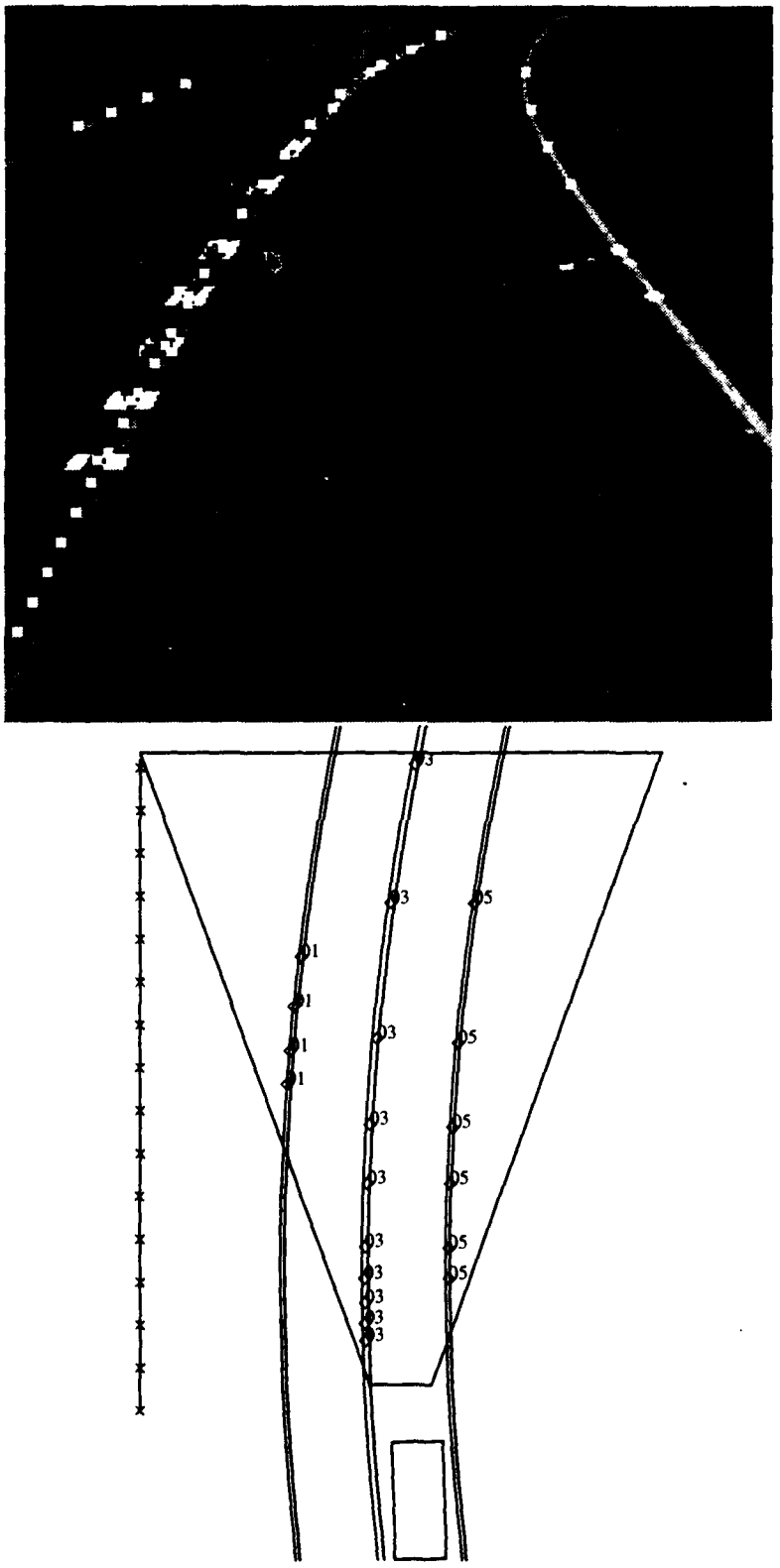


**Figure 21: Road image with detected feature locations (top) and recovered vehicle location and road curvature (bottom)**

the road model, standard statistical fitting techniques can be used to recover the spine arc parameters of *curvature*, *heading*, and *spinetrans*. YARF uses either standard least squares fitting or least median of squares fitting. The next subsections discuss in more detail the errors introduced by making the approximations used to derive this model, giving an analysis of the magnitude of the errors introduced.

### 4.3.1 Approximating a circular arc by a parabola

Consider the equation of a half circle centered at the origin, $x = \sqrt{r^2 - y^2}$. This can be expressed as a series $x = c_0 + c_1 \times y + c_2 \times y^2 + c_3 \times y^3 + \ldots$ Performing the binomial series expansion to solve for the coefficients results in the solution $c_0 = r$, $c_1 = 0$, $c_2 = -0.5 \times r^{-1}$, and in general $c_n = (c_{n-2} \times (n-3)) / (n \times r^2)$. Ignoring terms beyond $y^2$ yields the parabola $x = r + 0.5 \times y^2 \times r^{-1}$. The change of sign of the $y^2$ term is due to the curvature convention used by the Navlab controller (see Figure 8). Introducing translation in $x$ simply changes the interpretation of the constant term of the series from $c_0 = r$ to $c_0 = r + x_{center}$. Translation in $y$ makes the coefficient of the $y$ term in the series nonzero by substituting $y' = y - y_{center}$ into the parabola equation above.

The axis of the fit parabola is implicitly the $y$ value around which the series expansion is being made. The further data points lie from that axis, the greater the divergence between the estimate of arc curvature and the actual curvature. This effect is shown in Figure 23. The graph plots estimated curvature vs. the angle subtended by the data for an arc with unit curvature. Twenty data points were generated evenly spaced along the specified section of arc. The curve marked with diamonds shows the case where all the data lies above the $x$ axis. The curve marked with stars shows the case where the $x$ axis passes through the center of the data. Increasing the angle subtended by the data corresponds to holding the radius of curvature of the arc fixed and increasing the size of the local map window, or to holding the size of the local map window fixed and decreasing the radius of curvature of the arc.

As can be seen from the graph, as the fraction of the arc circumference spanned by the data increases, the error in the estimated curvature rises much more slowly when the $x$ axis passes through the center of the data. Since the model equations constrain the axis to be parallel to the $x$ axis, it suffices to rotate the data so that the $x$ axis is perpendicular to the predicted arc tangent at the mean $y$ value of the data points.

### 4.3.2 Translating data points perpendicular to the Y-axis

YARF tracks multiple features from the road model. Data points from features offset from the spine have to be translated to lie on the spine in order to combine the locations from the different features into an estimate of the spine arc parameters. The translation is made parallel to the $x$ axis rather than perpendicular to the (unknown) feature tangents in order to keep the problem linear (see Figure 24 below). The error in the $x$ coordinate of the corresponding spine point introduced by this approximation is $error = (x - x_{center}) - offset + \sqrt{(x - x_{center})^2 - offset^2 - (2 \times offset \times radius)}$.

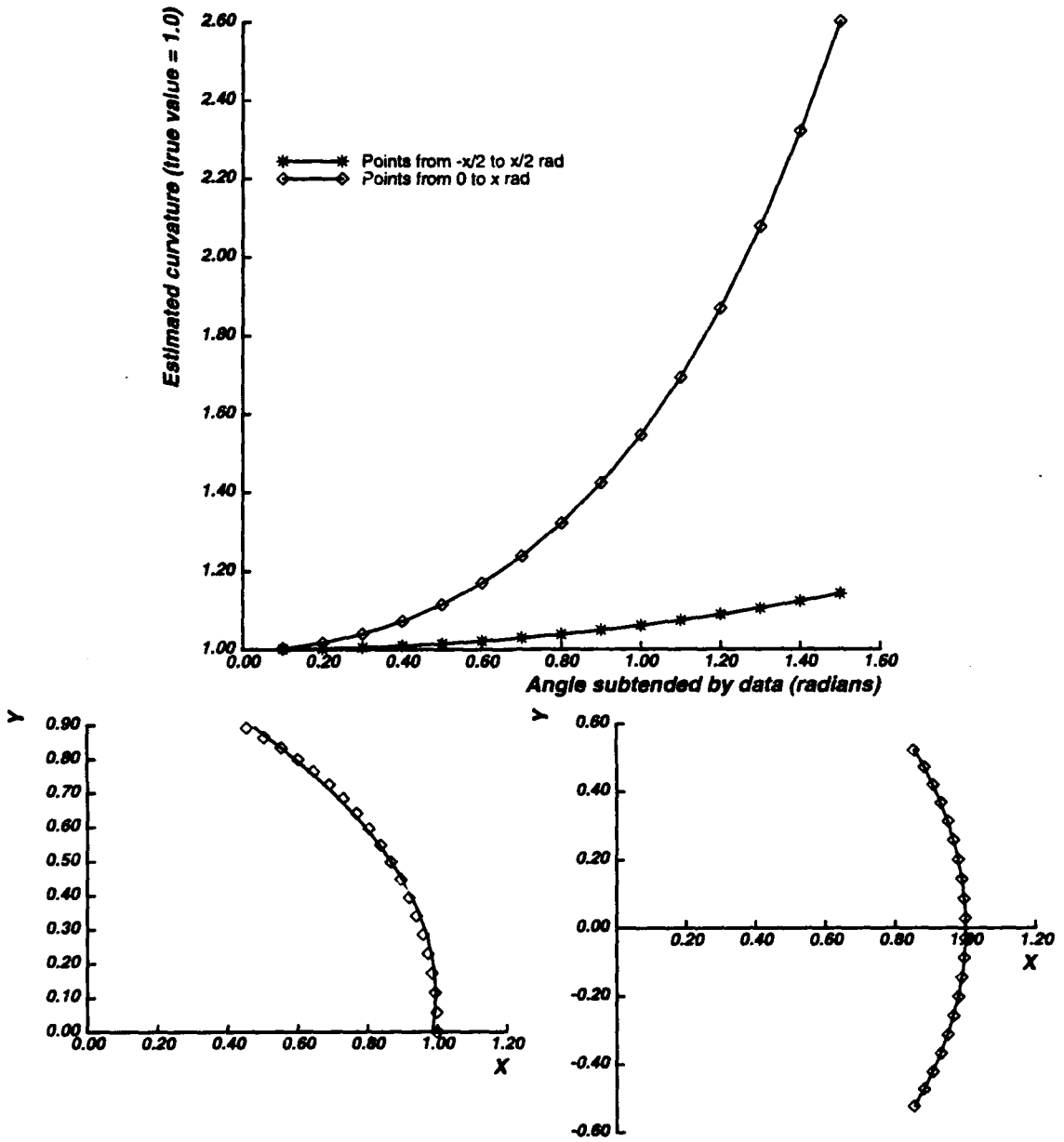



**Figure 23: Effect of coordinate frame on the estimate of arc curvature using the parabolic approximation, showing estimated curvature for cases in which points all lie above the *x* axis (bottom left) vs. cases in which *x* axis passes through the center of the data points (bottom right)**

The magnitude of this error is also dependent on the coordinate system chosen for the fit. Again, rotating the data so that the *x* axis is roughly perpendicular to the predicted road at the mean *y* value of the data spreads the error more evenly among the points and reduces the size of the error for the points with larger *y* values.

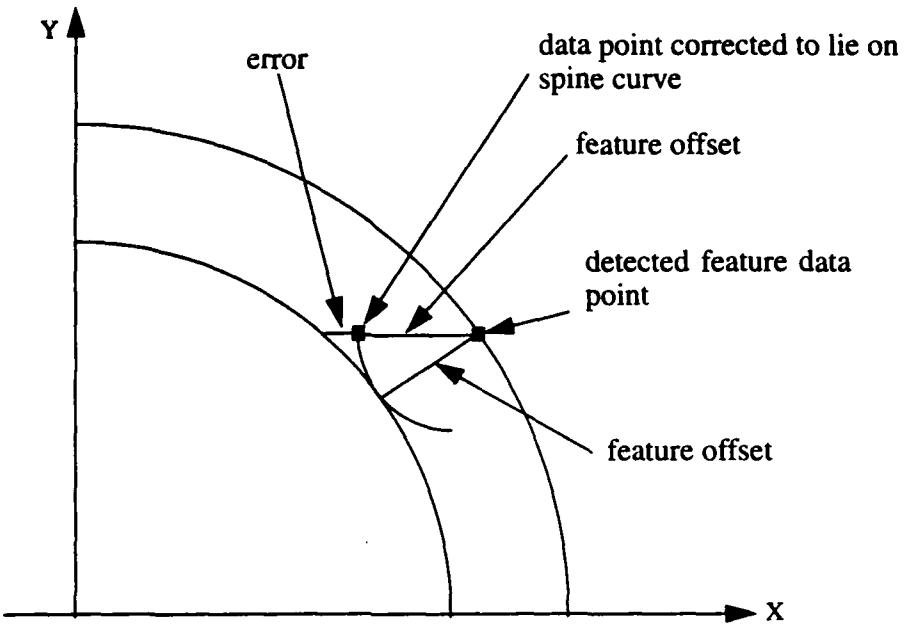



**Figure 24: Error introduced by translating points to the road spine parallel to the X axis**

In the extreme case all of the data comes from a single feature. This can happen in situations such as driving through a tight curve with a fixed camera, where the lane edge on the inside of the curve is typically not visible. In such a case the estimated curvature will be that of the visible feature rather than that of the spine, as the approximation would then be equivalent to shifting the feature arc horizontally by the feature offset. When the data points are balanced around the spine the effect is minimized. This makes the effect highly dependent on the specific geometry of the current situation. Unlike the error due to the series approximation, the error introduced by this approximation cannot be cleanly analyzed in the general case.

In the special case where all the data comes from a single feature offset from the road spine, the relative error is $error_{radius} = (radius_{spine} - radius_{feature}) / radius_{spine}$. Since $radius_{feature} = radius_{spine} + offset_{feature}$, $error_{radius} = -offset_{feature} / radius_{spine}$. Thus, tracking a single feature offset 3 meters (roughly 10 feet) from the road spine on a curve with a spine arc radius of 30 meters will introduce a 10% error into the estimate of the spine arc radius.

### 4.3.3 Evaluation of approximation errors: Simulation results

Simulations were run in order to provide quantitative estimates of the errors introduced by the approximations described above, and to show the importance of fitting the data in a "natural" coordinate system in which the $x$ axis is perpendicular to the road tangent at the mean $y$ value of the data points. We call such rotation of the data before model fitting *virtual panning*. The camera and road geometry models from an actual Navlab run were used to generate synthetic road images of specified curvature, and YARF was run on the synthetic images to gather data on the difference between the estimated road shape and the actual road shape.

The simulated vehicle drove 2 meters between images, keeping centered in the lane with the rear axle perpendicular to the spine curve. The simulator was set up to use the same road and camera models to generate the images and to backproject and fit the data, and the image data is idealized. This eliminates sources of error other than the approximations described above. After allowing the simulation to run for 10 frames to allow the system to settle into a steady state, the fits from the eleventh to twentieth frames were averaged and compared to the known model.

The error measure chosen was distance from the true lane center to the estimated lane center at a given distance along the estimated lane center arc. This error was plotted for distances starting at the rear axle of the vehicle and extending out to 40 meters along the estimated lane center. The front bumper of the vehicle is approximately 3.5 meters in front of the rear axle. The error measure is illustrated in Figure 25 below.

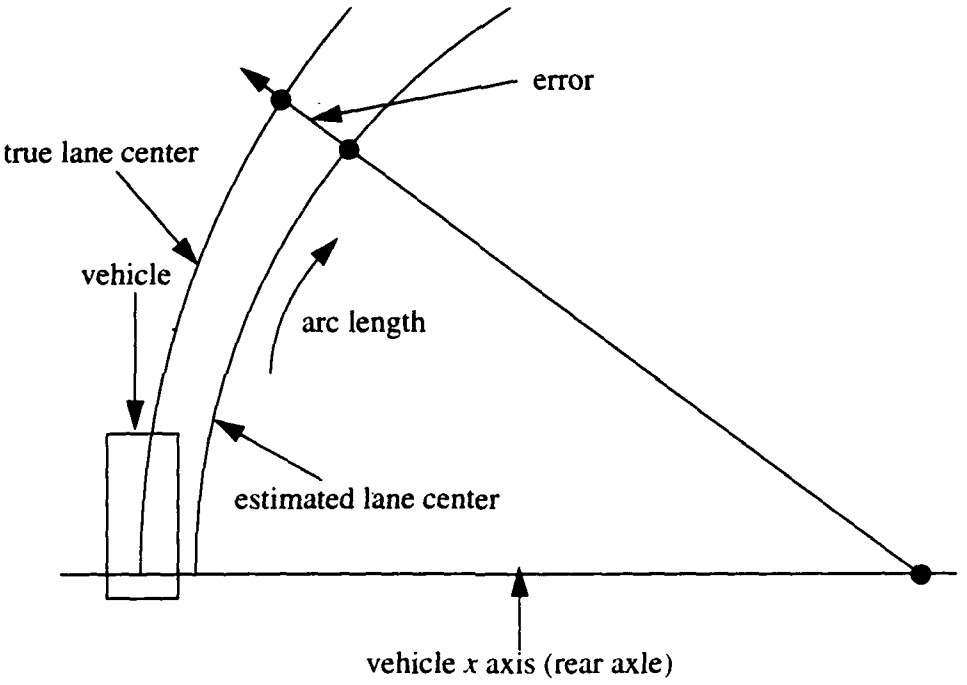



**Figure 25: Illustration of error measure for spine fit simulations**

Figure 26 shows the results for the first set of simulations, in which the parameter fits were done in the vehicle coordinate frame. Note that in all cases the error in the range of 5 to 15 meters from the rear axle is very small, staying within a foot of the actual lane center. As the curves become tighter the errors increase dramatically for distances greater than 20 meters along the estimated lane center. The asymmetry of the results for the positive and negative curvature cases is due to the pan of the camera and the offset of the camera from the center line of the vehicle, which results in different visibility of the left and right lane markers when the vehicle turns left than when it turns right.

Figure 27 shows the results for the second set of simulations. In this set of simulations the data was rotated so that the $x$ axis was perpendicular to the predicted road at the mean $y$ data value. Notice that the magnitude of the error is kept under 80 cm. at all distances out to 40

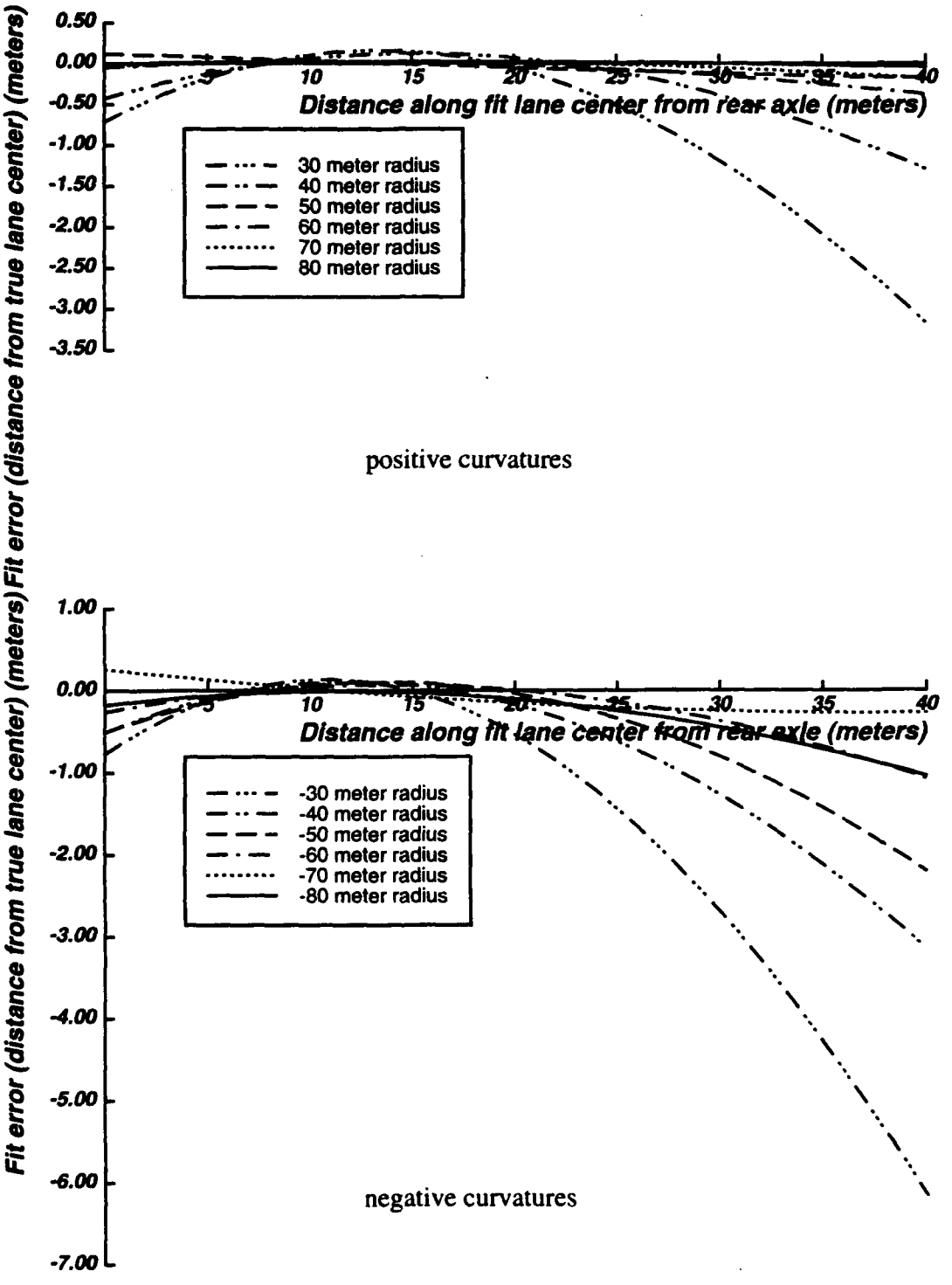



**Figure 26: Fit error, fits done in vehicle coordinate system**

meters along the estimated lane center, and for all radii of curvature down to +/- 30 meters. This shows the improvement in fit accuracy achieved by rotating the data into a "natural" coordinate system.

## 4.4 Parameter estimation by Least Median of Squares fitting

### 4.4.1 Robust estimation: terminology and the LMS algorithm

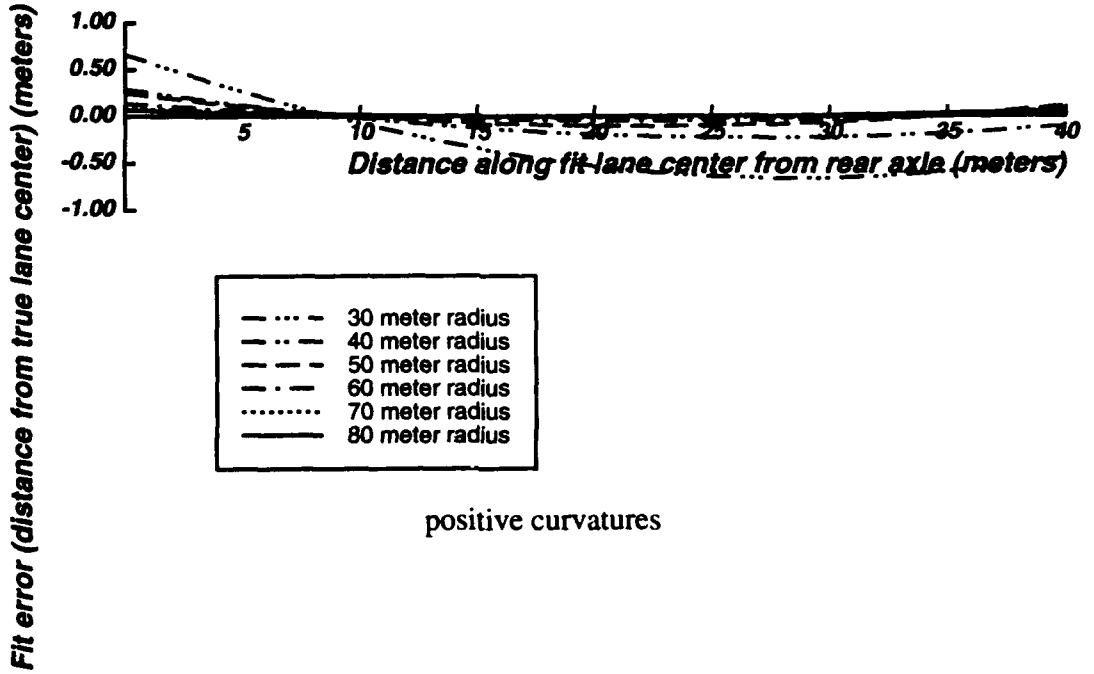
Data can be contaminated by observations which do not come from the process whose parameters are being estimated. Such observations are called *outliers*. Their presence in a data set can result in parameter fits that are grossly incorrect when standard least squares techniques are used as estimators. Outliers pose a particular problem for the YARF system. They will arise when there is a false positive response from a tracker, or when the road features deviate from the stripe model. Because the tracker windows are placed at the predicted road position, outliers will not be random and may pull the fit incorrectly towards the predicted road location and away from the actual road.

There are two approaches to making estimation insensitive to outliers. The first, *outlier detection*, attempts to identify the contaminating data points and eliminate them before performing a standard least squares fit to the remaining data. A good survey of techniques for outlier detection can be found in [3]. The second approach, *robust estimation*, attempts to design estimators which are less sensitive to the presence of outliers than standard least squares estimation.

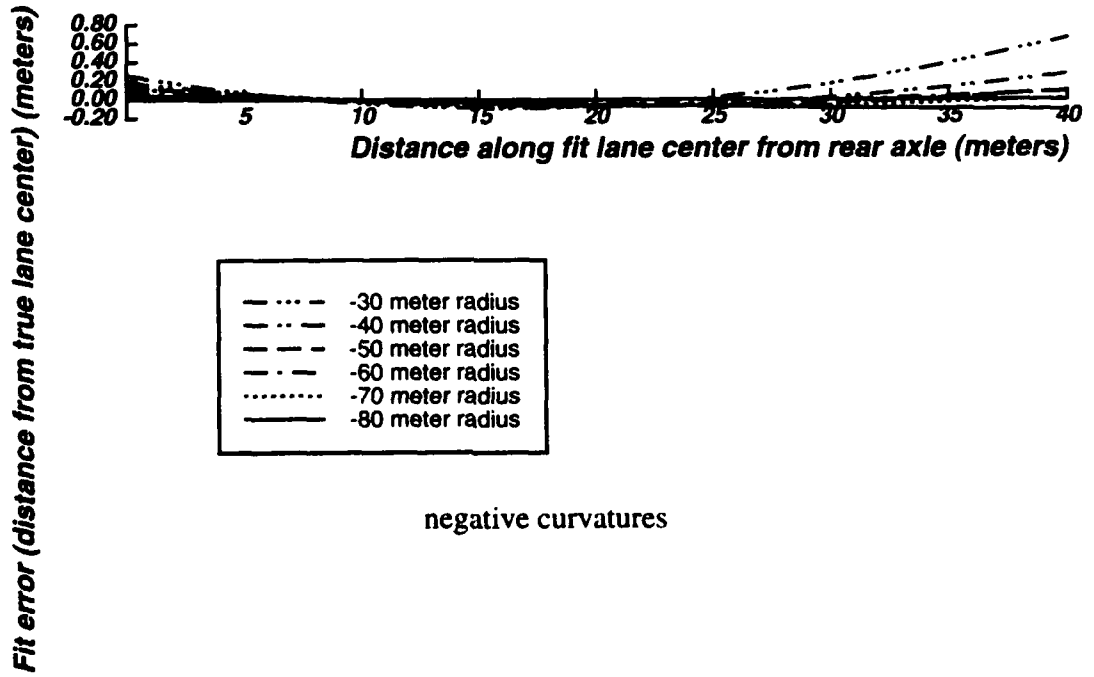
An increasingly popular robust estimation techniques is called *Least Median of Squares* (or LMS) estimation [43]. Consider the linear system $y_i = \beta x_i + \varepsilon$, where $\beta$ is the vector of parameters to be estimated, and $\varepsilon$ is a noise term. Standard least squares finds the estimate $\hat{\beta}$ which minimizes $\sum r_i^2$, where $r_i$ is the residual $r_i = \hat{\beta} x_i - y_i$. Least Median of Squares tries to find the estimate $\hat{\beta}$ which minimizes $median(r_i^2)$. The case of fitting a line in two dimensions provides a simple geometric intuition for what the LMS estimate is. The LMS estimate is the line such that a band centered on the line which contains half the data points has the minimum height in $y$ (the dependent variable) (see Figure 28).

The computation of the LMS estimate is straightforward, and is similar to Bolles' RANSAC algorithm [16]. Random subsets of the data are chosen. The standard least squares estimate of the parameters is made for each subset, and the median squared residual of the entire data set is calculated for that estimate. The estimate which produced the lowest median squared residual is selected as the final estimate.

The *breakdown point* of an estimator is the smallest fraction of the data that needs to consist of outliers in order for the parameter estimates to be pulled arbitrarily far from the correct values. In the case of standard least squares, the breakdown point is asymptotically zero, since a single outlying observation can cause pull the fit arbitrarily far from the correct result. The breakdown point of LMS estimation is 50%, the maximum achievable.




positive curvatures




negative curvatures

**Figure 27: Fit error, virtual panning of data before fit**

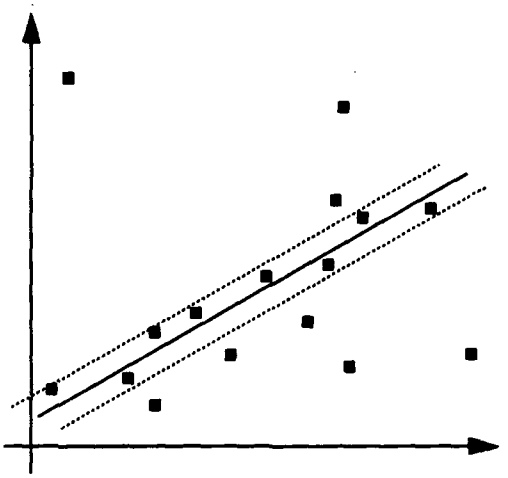


**Figure 28: Example LMS fit**

The *relative efficiency* of an estimator is the ratio of the lowest achievable variance for the estimated parameters (given by the Cramer-Rao bound) and the variance achieved by the estimator. In the case of LMS estimation the relative efficiency is $2/\pi = 0.637$. In practice, the variance in the parameter estimates can be reduced by using the value of the median squared residual to estimate the variance of the noise in the data, eliminating data points more than three standard deviations away from the initial LMS fit, and refitting the remaining data points using standard least squares.

Two phenomena complicate the task of identifying outlying observations. The first, *masking*, occurs when there are multiple outliers in the data which jointly influence the fit in such a way that their residuals do not look unusual. The second, *swamping*, occurs when outliers influence the fit in such a way that valid data points have suspiciously large residuals which make them appear to be outliers. Selecting a robust estimation technique which shows low sensitivity to masking is important in the YARF domain because outliers will occur near the predicted road location, and will therefore tend to influence the fit in a consistent way. LMS estimation shows little sensitivity to masking, another factor in its favor.

### 4.4.2 Examples of the effects of contaminants on estimated road shape

Contaminating data points can arise from a number of sources. The first source is false positive responses from the feature trackers. Figure 29 shows an example of such a situation. Snow obscures the white stripe marking the right edge of the lane. While the yellow stripe tracker correctly locates the double yellow line, the oriented bar tracker returns incorrect locations caused by texture in the snow. Because YARF searches for the features in small windows at the predicted feature location, the locations of these outliers are highly correlated with each other, producing masking. These incorrect data points pull the least squares fit away from the correct value, as shown by the rightward trend in the reconstructed lane center at farther distances.

Figure 30 compares the results of applying least squares and least median squares estimation to data from the area shown in Figure 29. The least squares result, shown on the left, splits the fit error among all the data points. As a result, the estimated road shape is incorrect, with

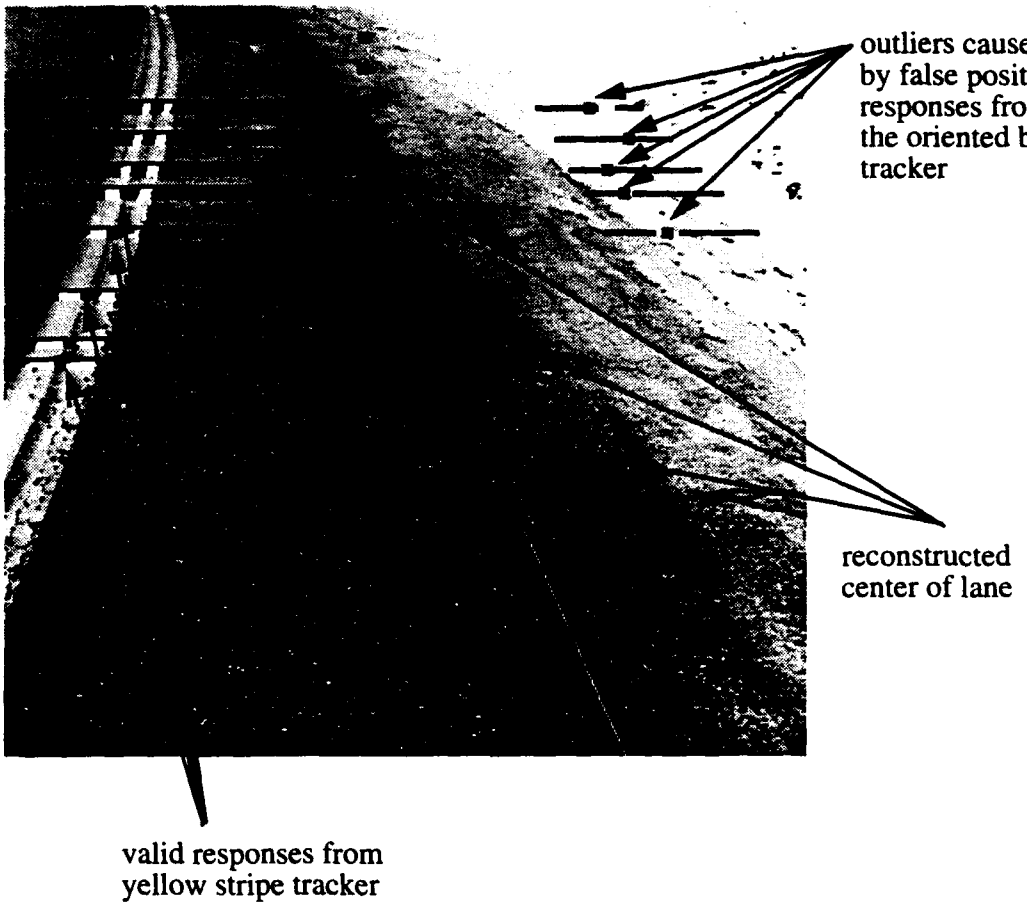



**Figure 29: Outliers caused by false positive tracker results**

the reconstructed lane located between the valid data points from the double yellow line on the left edge of the lane and the invalid data points from the snow on the right side of the lane. The least median squares result, shown on the right, correctly ignores the outliers on the right side of the lane and places the valid data points from the left lane marking within the reconstructed double yellow line

False positive responses from the low level image segmentation techniques are the most obvious source of contaminating data points, but changes in road appearance present another source. Exit ramps are an example of this type of source of outliers. While the left edge of the lane continues in a way consistent with the previous lane curvature, the right edge of the lane veers off to the right. Least squares will try to split the error between the points from the left and right lane edges, and may track the exit ramp by mistake as a result. This problem has been observed in a number of other road following systems, including VITA [17] and ALVINN ([40], Section 5.1.6). Because least median squares selects a fit which ignores points that are inconsistent with the majority of the data, it can avoid the influence of points on the right lane marking as it veers off for the exit ramp.

In order to demonstrate this, both least squares and least median squares estimation were performed on a sequence of images tagged with the associated vehicle positions that had been taken on a highway near Pittsburgh. Figure 31, Figure 32, and Figure 33 show the results of the two estimation methods for three of the frames in the sequence. In Figure 31,

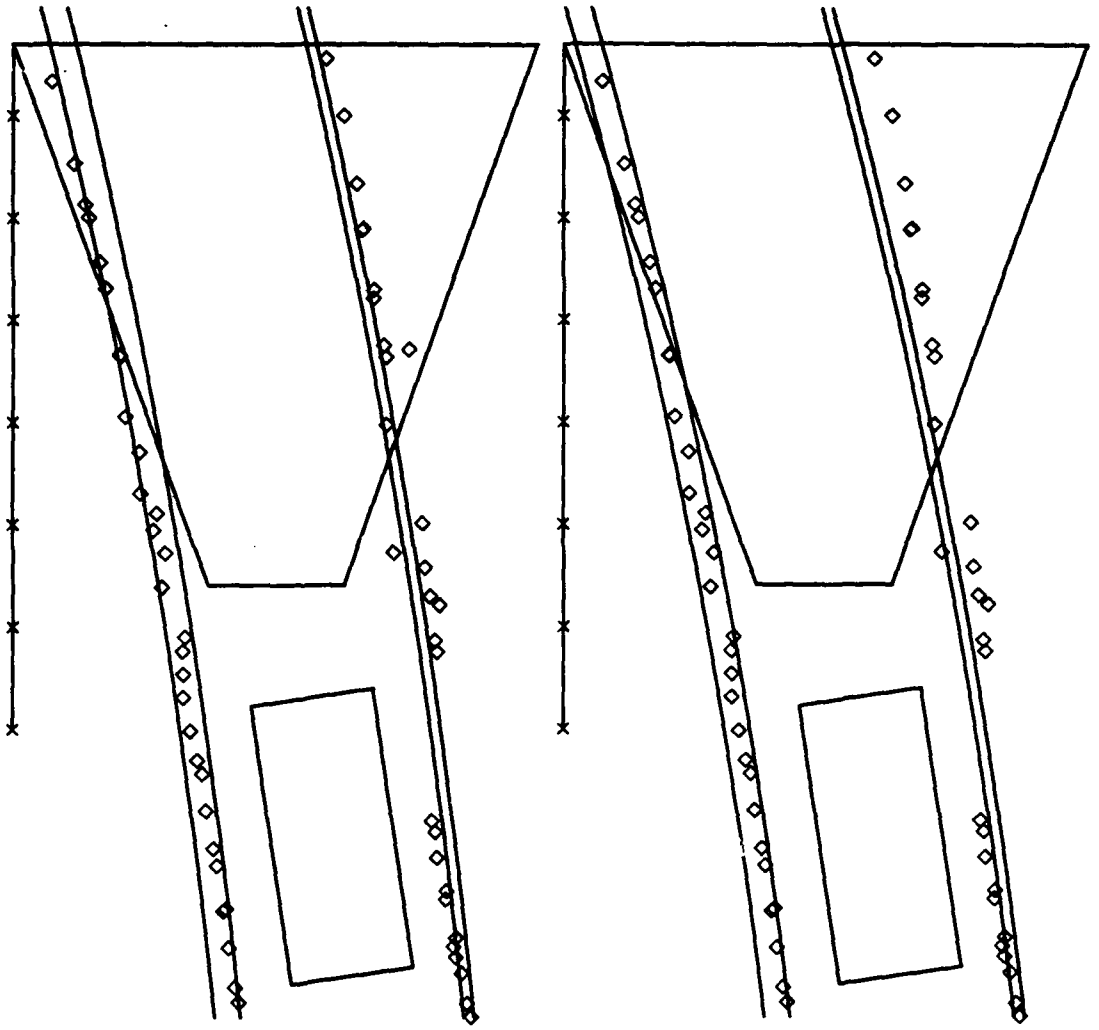


**Figure 30: Least Squares (left) and Least Median Squares (right) road fits to data with outliers from snow covering the right lane marker**

showing the results for frame 6 of the sequence, the least squares and least median squares techniques produce essentially identical fits. In Figure 32, frame 12 of the sequence, the least squares result has placed the estimate of road position between the points from the left and right edges, and as a result has incorrectly locked onto the exit ramp. This can be seen from the lack of points from the left lane edge near the top of the camera field of view, and the points from the right edge of the exit lane in the same area. Least median squares, on the other hand, correctly detects that the right lane marker is veering away in a manner inconsistent with the bulk of the data. It ignores those points (which appear noticeably to the right of the estimated right lane marker position), and correctly estimates the lane curvature as a mild curve to the left, tracking the left lane marker. By the frame shown in Figure 33, frame 22, the divergence is obvious. Least squares has lost the left lane edge and is tracking the right edge of the exit ramp. Least median squares has correctly tracked the left lane edge and has picked up a single point near the top from the right lane edge marker where it reappears near the top of the image.

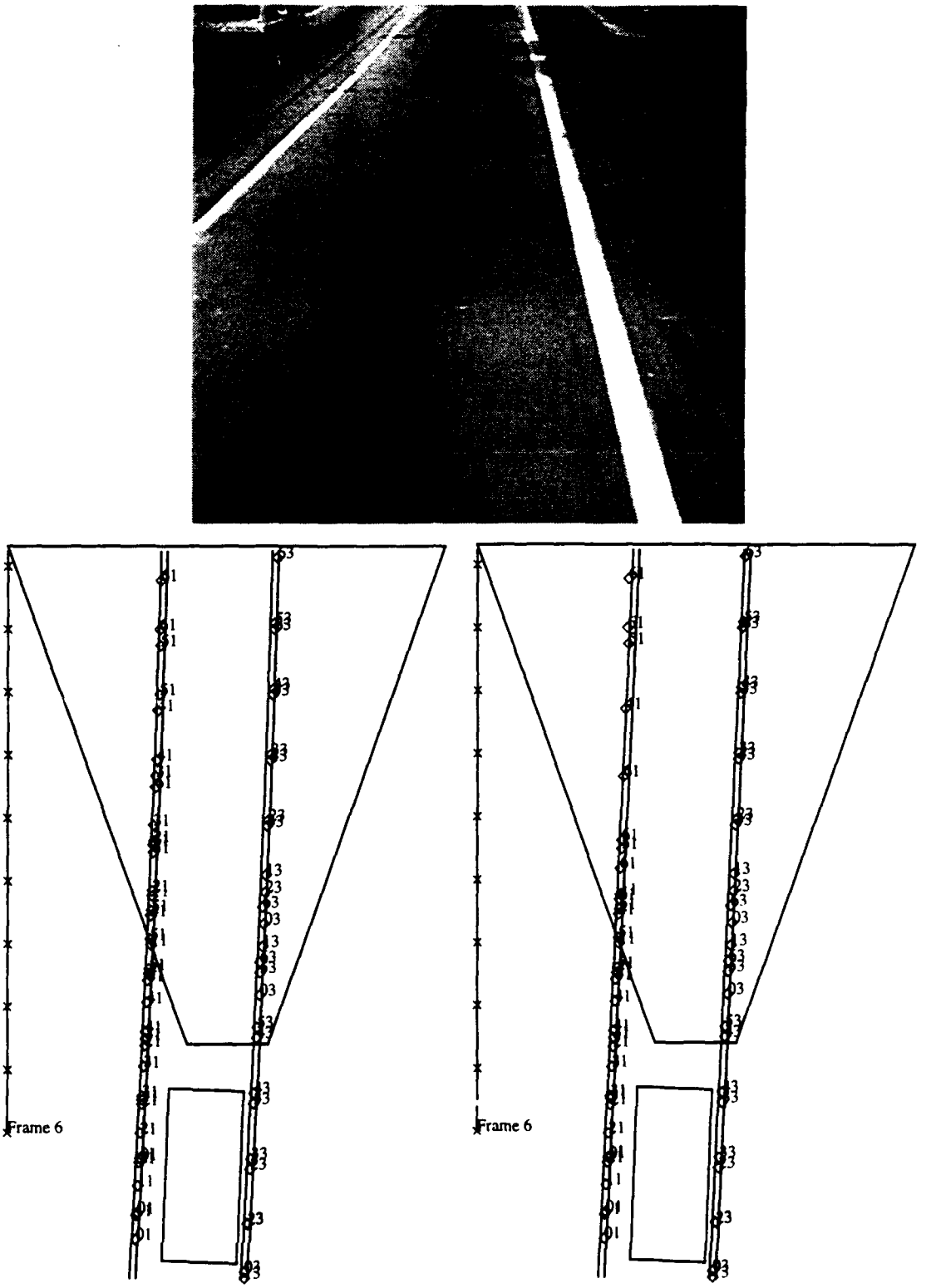



**Figure 31: Least squares result (bottom left) and least median squares result (bottom right) for frame 6 in exit ramp image sequence (top)**

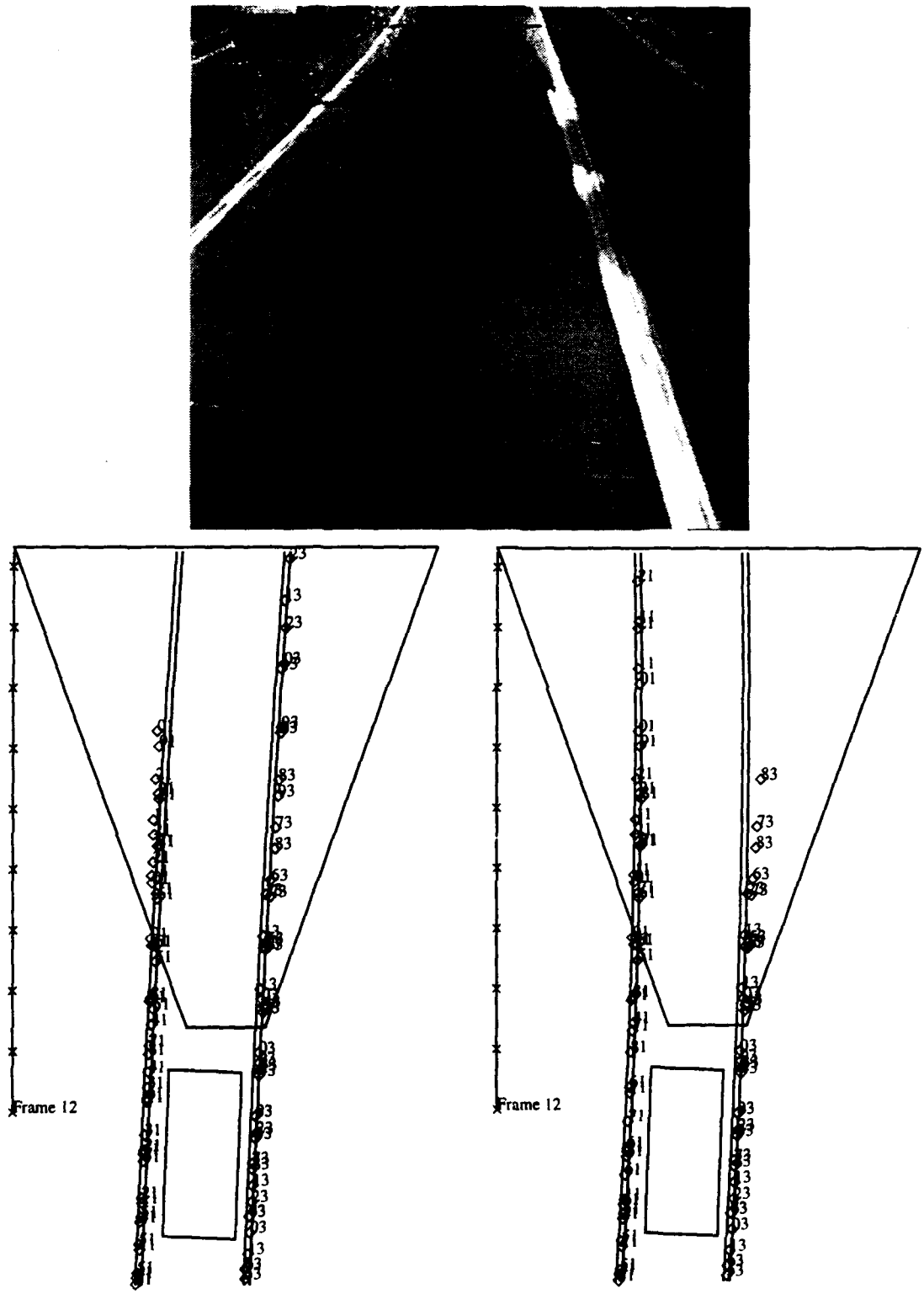



**Figure 32: Least squares result (bottom left) and least median squares result (bottom right) for frame 12 in exit ramp image sequence (top)**

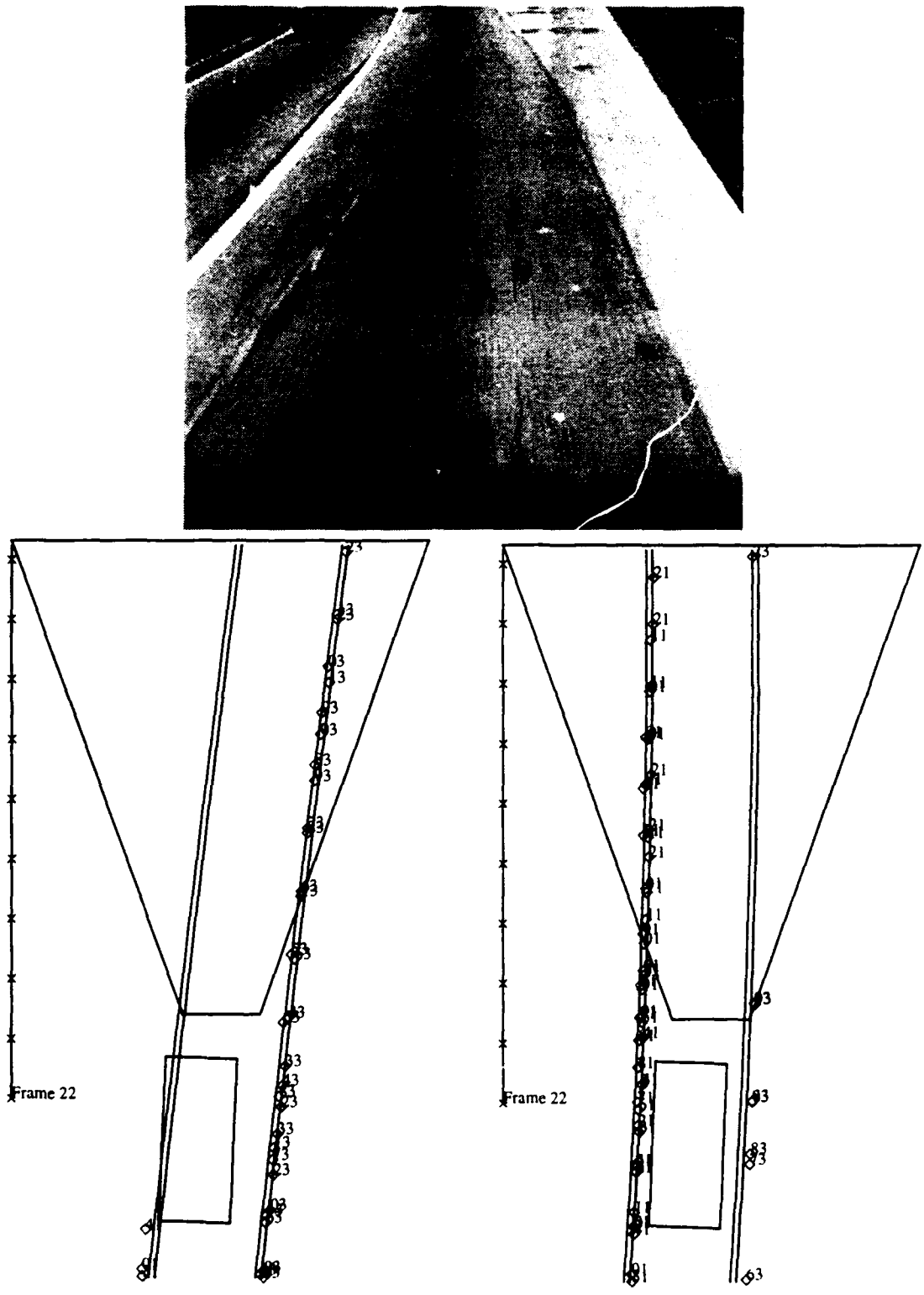



**Figure 33: Least squares result (bottom left) and least median squares result (bottom right) for frame 22 in exit ramp image sequence (top)**

Both lane edges were marked by solid white stripes in the example shown above. In many cases the left lane edge will be marked by a broken white stripe, with the result that data from that edge of the lane will be more sparse. This will interact with the median square residual criterion to make it more difficult, if not impossible, for least median squares to handle such cases. This suggests a direction for future research to develop variants of the standard least median squares algorithm which incorporate knowledge about the data to handle such imbalance. Also, the lower relative efficiency of the least median squares estimator introduces larger amounts of noise into the system's estimate of road curvature. Some smoothing of the curvature estimates may be necessary.

In addition, it is possible that the exit ramp may be a better continuation of the data than the lane. Currently YARF does not deal with this possibility, and uses the LMS estimate to chose the best continuation of the data as the correct answer. In order to handle the general case, it would be preferable to explicitly tag data points as outliers based on their residual error relative to the LMS fit. The local map could then be scanned for groups of adjacent outliers which might indicate a shift in lane marking position, and higher level information about the road could be brought to bear to determine how to respond to the situation. Such an extension would be very similar to the existing techniques used in YARF to detect changes in lane structure and approaching intersection, as described in Chapter 5.

## 4.5 Conclusion and future work

YARF uses a non-linear model of road shape. YARF fits a low-order polynomial function to the feature data in order to be able to use a linear estimation technique. YARF then determines the parameters of the non-linear shape model by assuming that the coefficients of the polynomial are related to them through the corresponding terms of the power-series expansion of the model. In the case of YARF, the spine is assumed to be a circular arc, and a parabola is fit to the data. In the VaMoRs [12] and VITA [18] systems, the spine is assumed to be a clothoid, and a cubic polynomial is fit to the data. The experimental results presented in this chapter show how the magnitude of the errors introduced by this method depends on the choice of coordinate system in which the fit is performed, and demonstrate that rotating the data into a "natural" coordinate system significantly reduces the size of the errors.

The use of least median squares to perform the shape parameter estimation was described. Examples were shown demonstrating the immunity of LMS estimation to contaminants in the feature data in situations where a least squares based system would fail. While uneven representation of features in the data set may make it impossible for LMS to handle all such situations, YARF improves on existing systems and opens up avenues for addition research on robust estimators which can handle data imbalance.

The road model chosen for YARF, which uses the flat-earth assumption and a circular spine arc, does not represent the current state of the art. Extending the road model to use a clothoid spine curve under the flat earth assumption would be straightforward, and would involve adding a cubic term to the series approximation to the spine curve. Extending the road model to include vertical curvature would be more difficult. Extracting the feature locations by a technique which directly detected their 3D position would allow easy construction of a 3D local map and allow the decoupling of the estimation of horizontal and vertical road curvature. Either stereo or fusion of color and range data could be used to provide such 3D feature data. This would eliminate the need to simultaneously estimate the height of the data

points along with the road shape. As a result, the horizontal and vertical curvatures could be estimated by fitting a cubic polynomial to the projections of the data onto the $X$-$Y$ plane and $Y$-$Z$ planes.

# 5 Map based navigation

The algorithms described so far permit YARF to track a segment of road which has a constant feature cross section. Lane markings terminate at intersections, and lane structure can change as the road changes width, or as a turn lane appears approaching an intersection. A versatile road following system needs to be able to detect and react to such changes in road structure.

This chapter describes how YARF detects and reacts to changes in road feature cross section which are predicted by the system's world model for the mission it is performing. Locations where the feature trackers fail to detect expected features are examined to determine if the pattern of failures is consistent with the behavior of the road markings at the end of the current road segment. If so, then the mission map is used to predict the location of features on the next road segment. Feature trackers are applied at these locations to verify or refute the predicted change in lane structure.

This results in a very different style of map-based navigation than that used in most previous autonomous navigation systems. The detection of approaching landmarks is triggered in a bottom-up manner through the detection of environmental cues. This is in contrast to the usual top-down approach, in which perception to detect approaching landmarks is triggered based on proximity to the landmark in a world model defined in terms of a global coordinate system. Landmark locations are much less constrained, placing a larger burden on perception. Detection of a particular landmark is a process extending over multiple image frames, rather than a one-shot process.

## 5.1 Global coordinates considered harmful

Related previous work includes two types of research. The first is general research into using landmark information in maps as an aid to robot navigation. The second is research into using a combination of map information and image features to locate and navigate intersections. Most existing systems in these two categories share a particular implicit model of mission execution and of the nature and function of landmarks.

The system built by Fennema [15] for the Harvey robot at the University of Massachusetts uses landmark descriptions stored in the world model for two functions. In the first, action-level perceptual servoing, landmarks are used to monitor the robot's performance of primitive actions such as "steer 0.34 radians left" or "move 2.56 meters". This is done to reduce the magnitude of error in command execution compared to open-loop execution. The second function, plan-level perceptual servoing, uses landmarks as milestones in the execution of a specified mission plan. As the system determines that it should be approaching a milestone landmark it begins looking for the landmark using its perception capabilities. Once the landmark has been detected the system servos the vehicle to correct any error between where it thought it was and its actual location in the world.

Thorpe and Gowdy [50] define the concept of an *annotated map* as a way of organizing data for autonomous navigation. *Annotations* consist of a header, which specifies the type and location associated with the annotation, and a data field, which can contain arbitrary data. The location assigned to an annotation can be either a point, a line segment, or a polygon (defined in world coordinates). Queries retrieve all annotations of a given type within the

specified query polygon. A *trigger* is a special kind of annotation. If the vehicle's path crosses the location of a trigger (generally a line segment or polygon), a message containing the data associated with that trigger is sent to a module specified in the header of the trigger annotation. One of the purposes of the annotated map formalism is permit off-line specification of decisions for which there are no autonomous planning capabilities in place. This is done through the use of triggers to specify choice of perception module, route at decision points, and so on.

Kushner and Puri at the University of Maryland [32] describe recognition of intersections using a mixture of map data and extracted image features. Their system begins by extracting the road boundary in an image. Points are sampled on the boundary and backprojected onto the ground plane. These points are then matched to road edge segments from the world map.

The Sidewalk II system at Carnegie Mellon [19] similarly matches image and map data to navigate a network of campus sidewalks. A simple histogram thresholding technique is applied to the blue band of a color image to classify pixels as road or nonroad. The boundary of the road region is then approximated by a polygon. Edges of the road region are matched to edges predicted from the map data. This allows the system to detect intersections modeled in the map and navigate through them.

These systems share a style of navigation in which actions, including the execution of perception routines, are triggered by the vehicle approaching points or entering regions which are defined in terms of a global coordinate system. That is very different from the style of navigation implied by the type of directions humans give each other for navigating on a road network. Consider this example, describing the route from the Carnegie Museum in Pittsburgh to an exhibit at another location:

> "From The Carnegie parking lot, go north on Craig Street to the first light and turn left (west) on Fifth Avenue;... Turn left on Craft Avenue, then right on Forbes Avenue to the on-ramp of the Boulevard of the Allies. Once on the Boulevard, follow the Three Rivers Stadium signs to I-376 West (the Parkway). From the Parkway, take I-279 North across the Fort Duquesne Bridge toward the North Side; stay in the left lanes. Exit at Three Rivers Stadium and turn right at the first light onto East Allegheny Avenue. At the third light, turn right onto W. North Avenue; at the fifth light turn left onto Federal Street. Turn left at the fifth street onto Jacksonia Street (one-way); the entrance to the parking lot is at 505 Jacksonia, about five blocks on the left, opposite Garfield Street." (directions to the John Cage installation for the 1991 Carnegie International)

A key feature of such directions is that the model they provide has no metric information of any kind. Landmarks are counted ("...turn left at the fifth street..."), but they are not used to locate the vehicle in some global Cartesian coordinate system, in strong contrast to the way landmarks are used in most current autonomous navigation systems. The style of navigation implied by such directions is one in which actions are triggered by the recognition of landmarks which are defined by the sequence in which they are encountered, and not by their location in a world-based coordinate frame.

Kender and Leff [26] provide a theoretical analysis of the complexity of performing this style of navigation. They model the task of landmark and sensor selection in a scenario where a robot is travelling along a one-dimensional environment such as travelling along a

single road or corridor. The environment has a set of landmarks along its length. These landmarks are defined only by their appearance and the sequence in which they will be encountered. The system is assumed to have multiple perception routines capable of detecting different (possibly overlapping) sets of landmarks. They show that selecting which landmarks to detect and which perception modules to use to detect them in order to minimize the cost of perception involves solving an NP-complete problem.

The ALVINN system [40] provides a connectionist framework for performing this style of data-driven navigation. Different neural networks are trained to detect different types of roads or landmarks. The output of each network is compared with the ideal pattern of output activation the network was trained to generate to measure how typical a given input image is of the type of feature the network was trained to recognize. Recognition of landmarks is triggered by changes in the confidence of the various networks. As an example, when the system approaches an intersection, the confidence of the road following network drops sharply due to the change in road appearance at the intersection. The system can then consult its map to determine which way to navigate through the intersection.

YARF uses a more traditional symbolic approach to detecting intersections in a data-driven fashion. The techniques used eliminate the need for a detailed description of the shape of the road segments between intersections. They also eliminate the need for an accurate estimate of the vehicle location in a global coordinate system. Estimates of vehicle motion between the capture of successive images are used to register data from multiple image frames. The errors in estimating vehicle motion over the distance between successive images are typically very small. The accumulation of error over time in the estimate of global vehicle position does not matter in YARF, and can be ignored.

## 5.2 YARF's road map

YARF models the network of roads to be traversed during a mission as a graph, with each edge in the graph representing a segment of road with constant lane structure. These segments are modeled by generalized stripes as described in Chapter 2. While the feature cross section of each segment is represented in the map, the shape of the spine curve is not. The spine curvature is determined from the detected feature points, as described in Chapter 4.

Each node in the graph represents either a change in lane structure or an intersection. In the case of changes in lane structure, YARF assumes that the spine tangents and curvatures match at the transition between the two stripes. In the case of intersections, the map provides the relative position and orientation of the end of each stripe at the intersection. These are defined in a coordinate system local to each intersection. As the vehicle drives along the road, information from the feature trackers is used to locate the end of the current road segment. This determines the position of the vehicle in the local coordinate system attached to the intersection being approached. By detecting the locations of the ends of road segments in a data driven fashion, the need to define the road network in terms of a global coordinate system is eliminated.

In addition to information about road topology and intersection geometry, the map also provides control information to specify decisions which would be made by the tactical level planning routines in a fully integrated system for autonomous road navigation. Each road

segment has annotations which specify which features in the stripe cross section to track; which feature trackers to use to detect those features; and which lane the vehicle should use in that segment. Figure 34 summarizes the information represented in YARF's global map.

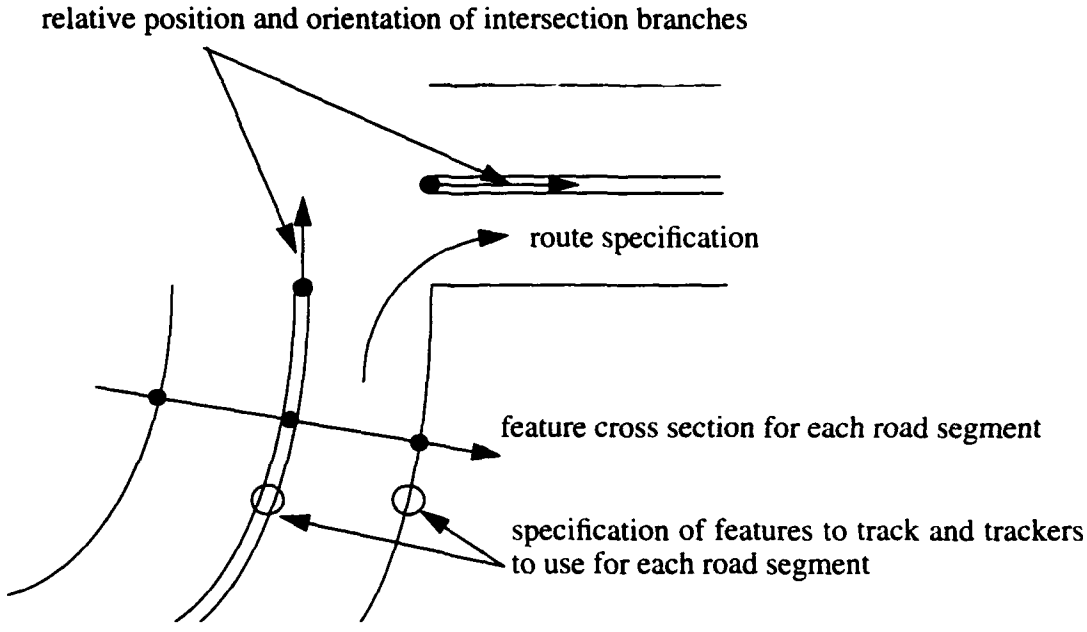



**Figure 34: Description of the global map information used in YARF**

The global map is static, and is generated off line. Figure 35 shows an example of the YARF map for a simple mission. The road at the bottom changes lane structure to include a left turn lane as the road approaches the intersection at top. The relative locations and orientations of the branching roads at the intersection are shown as defined in the map. Driver's aids such as the *Travelpilot*$^{TM}$ system [7] can be expected to provide information about the topology of the road network, and at least some crude information about intersection geometry. Such systems are unlikely to include detailed information about the lane structure of the roads, however. As a result, research will be needed into techniques to extract lane structure from images without a prior model. Chapter 6 describes efforts along those lines carried out as part of this thesis.

In the annotated map system described by Gowdy and Thorpe, control annotations are triggered when the vehicle's estimated position falls inside a specified region in the map. In the YARF system, the transition to a new road segment is instead triggered when the behavior of the features observed by the system changes in the manner predicted by the map. Using the map shown in Figure 35 as an example, the double yellow line changes to a single white stripe at the transition from segment 1 to segment 2, and a new double yellow line appears to mark the left edge of the turn lane. By recognizing the end of the yellow stripe in segment 1, YARF can test for and verify the predicted lane markings in segment 2.

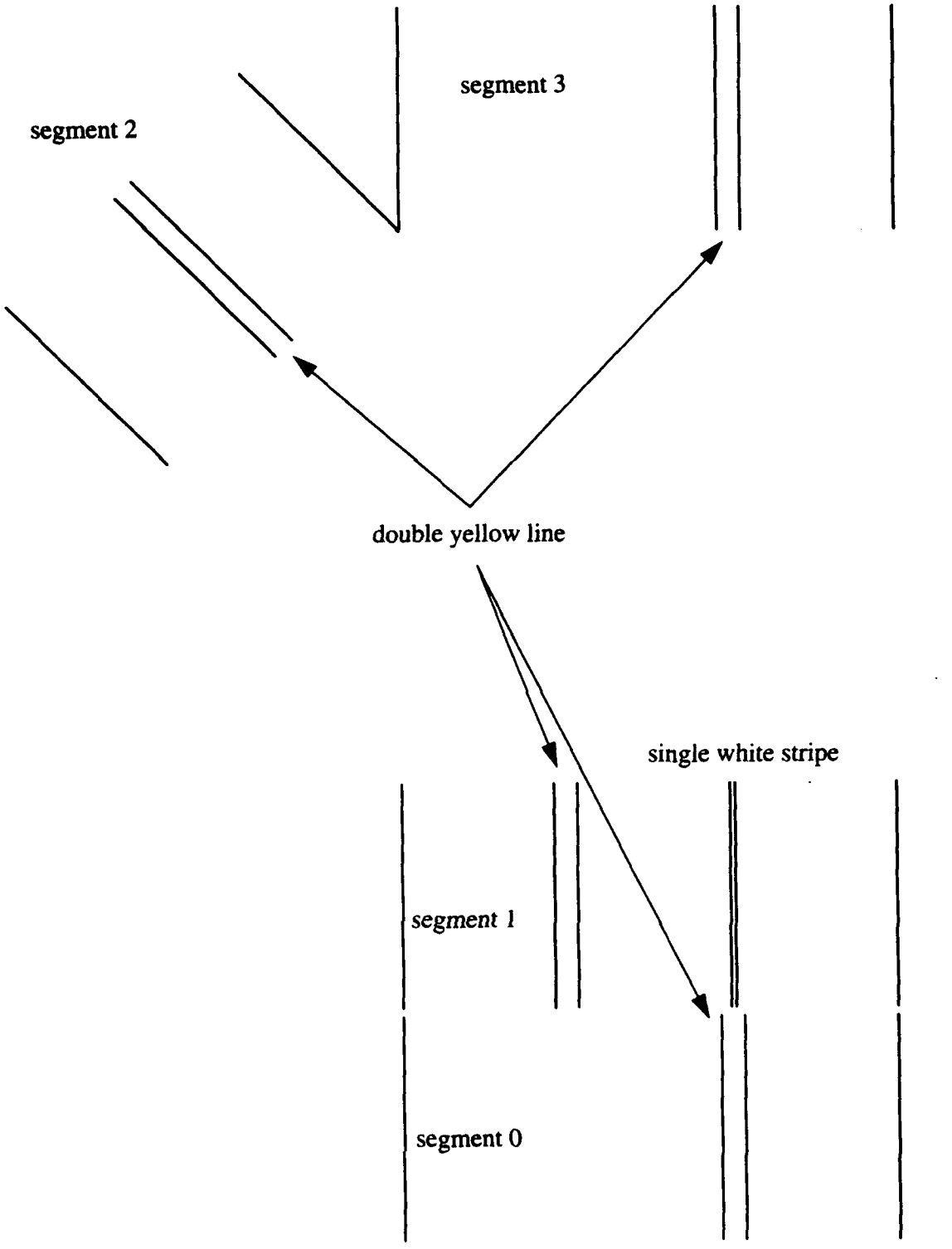



**Figure 35: Simple YARF map**

## 5.3 Hypothesizing intersections based on tracker failures

Expected intersections and changes in lane structure are detected through a combination of expectation generated by the map with data from the feature trackers. Individual point failures to detect features are examined to see if their pattern matches that predicted for the end of the current road segment in the map. If so, a hypothesis is generated that the vehicle is approaching the next stripe transition (lane structure change or intersection) along the mission route. Trackers are placed along the predicted location of features from the next road segment along the route. If the results of the trackers confirm the hypothesis, then YARF continues to track the new road segment as the vehicle travels through the segment transition. Otherwise, YARF attempts to continue tracking the current road segment.

There are four stages in intersection hypothesis generation. In the first stage, locations where trackers failed to detect predicted features are entered into a local map of the area around the vehicle. In the second stage, the updated estimate of road location in the current image is used to check whether any failures were due to mispredictions of feature location. In the third stage, the tracker locations for each feature in the local map are examined for runs of adjacent failures which may correspond to gaps in the continuity of the feature. In the fourth stage, the different features being tracked are examined for overlapping gaps which could signal the approach of the next intersection along the route.

### 5.3.1 Detection of tracker failures

As described in Chapter 3, YARF uses multiple image segmentation algorithms, each of which is specialized to robustly detect a particular type of road feature. The predicted road location is used to generate search windows at selected rows in the current image. The trackers then examine those search windows to detect the actual location of the predicted features. The trackers are designed to indicate when a search window didn't contain the expected feature as well as to provide the feature location when the search window does contain the feature. The hue-based tracker for yellow stripe detection described in Section 3.3.1 provides an example of how this is done. When the tracker algorithm detects that no pixels in the search window fell inside the specified section of color space, the tracker returns an error value indicating that the feature was not seen.

Tracker failures may be due to the actual absence of the feature, or they may be due to errors on the part of the segmentation technique. In order to reduce the possibility of erroneously hypothesizing the end of the current road segment, it is desirable to try to explain the cause of tracker failures when possible. Some classes of incorrect failures can be detected based on the local properties of individual search windows. YARF currently only checks for one common cause of false failures, image saturation. On bright, sunny days the limited dynamic range of the color camera used can result in images where some areas in shadow are black while other areas which are not in shadow are white. The tracker counts the number of pixels in the search window which are black (intensity $< 2$) or white (intensity $> 253$). If more than 90% of the pixels in the window are black or white, the tracker returns an error value indicating that the window was saturated due to camera dynamic range limitations. This allows YARF to disregard failures which can be explained based on image saturation in the window. By eliminating such tracker failures from further consideration, YARF can reduce the chance of triggering an incorrect lane structure change or intersection hypothesis.

### 5.3.2 Integration of tracker results in a local map

Section 4.2 described the process of constructing the local map. Data is integrated over multiple frames by transforming old data into a current vehicle-centered coordinate system and back-projecting new data onto the assumed ground plane. In addition to recording locations where features were detected within the search windows, locations where the features were expected but not found are also recorded.

### 5.3.3 Identifying gaps in individual features

Section 5.3.1 explained how some false tracker failures could be detected and explained based on a local property of the individual search windows, the fraction of pixels within a few grey levels of black or white. Other types of false failure cannot be detected based on local image properties. One class of such failures is due to mispredicted search windows. In this case, the failure to see the feature results from looking for it in the wrong location in the image. Once YARF has updated its estimate of road location and curvature, repredictions are made of feature locations for search windows where there were tracker failures in the current image. A failure is classified as a misprediction if the updated prediction of feature location doesn't fall within the original search window. By explaining these tracker failures, they can also be eliminated from further consideration.

Isolated tracker failures can occur due to badly worn or indistinct features or occlusion of the feature by leaves or snow or reflections off a puddle of water over the feature. YARF looks for support from multiple tracker failures in order to identify regions where a feature is absent. The local map stores the tracker detections and failures for each feature as a list sorted by $y$ value in a vehicle-centered coordinate system. This list of tracker results is scanned, and runs of tracker successes and failures are extracted, creating a list of feature segments and gaps. Failures classified in previous steps as caused by image saturation or search window misprediction are ignored during gap extraction.

Sections of the feature between the last detected point in a feature segment and the first failure in a gap are labeled as unknown intervals. So are the sections between the final failure of a gap and the first detection in a feature segment. These unknown intervals are explicitly represented due to the possibility that a feature may not be visible as the vehicle approaches an intersection because of limitations on camera field of view. As a result, assuming that the feature was present could cause a failure to hypothesize that the vehicle was approaching the intersection. This process of segmenting the feature data in feature segments, gaps, and unknown intervals is shown in Figure 36.

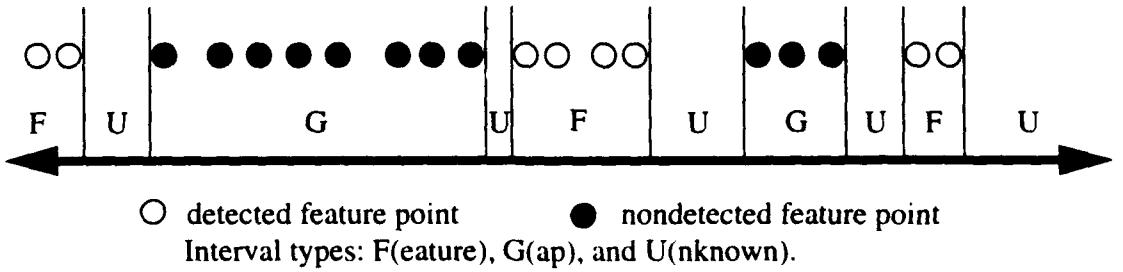



**Figure 36: Extraction of feature intervals from raw tracker data points**

Gaps with at least five tracker failures and spanning at least two meters are considered as well supported by the data, and are examined along with the unknown feature sections in the final stage of intersection hypothesis generation. Figure 37 shows an example of such a gap. The image on the left shows a section of road with a gap in the double yellow line. The overhead view on the right shows the local map generated by YARF during a run along this section of the road. The yellow-hue tracker failures in the gap are marked by asterisks, and the run of failures corresponding to the gap in the double yellow line is marked by brackets.

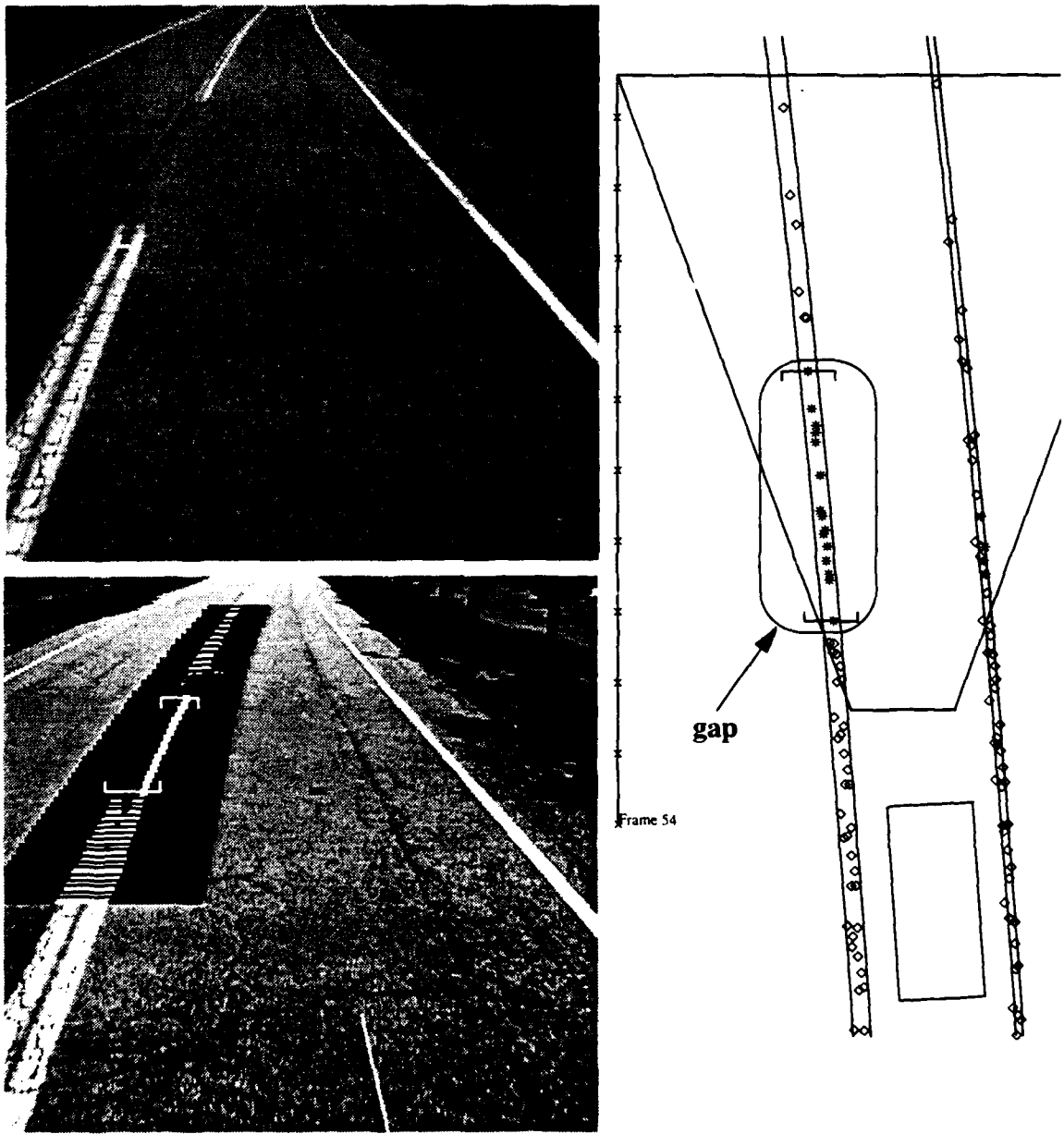



**Figure 37: Road scene with a gap in the double yellow line (top left); processed image showing tracker failures and extracted gap (bottom left); and view of local map from a Navlab run showing the failures (marked by asterisks) and extracted gap (bounded by brackets) (right)**

### 5.3.4 Hypothesizing the end of the current road segment

The previous stage of processing identified possible gaps in the individual features being tracked. The next step combines the gap information from all the features to determine if the end of the current road segment should be hypothesized. In order to do this, YARF examines the map to determine which features of the current road segment are expected to disappear at the next intersection. The interval information for those features is combined through a voting scheme to check if the end of the road segment appears to be approaching.

The voting scheme partitions the *y* values in the local map into bins with fairly close spacing to achieve good localization of the end of segment. Each bin has a count indicating the number of features which had a gap at the *y* value corresponding to the bin, and how many features had an unknown interval at that *y* value. YARF scans the intervals of the features which are expected to disappear at the end of the current stripe. Gap and unknown intervals vote for the bins they include. If the sum of the gap and unknown votes in a bin is equal to the number of features voting, then that bin is marked as part of a possible end-of-segment hypothesis. Runs of bins which are marked as possible end-of-segment hypotheses are extracted. The *y* value of the start of the longest run is returned as the hypothesized end of the road segment (see Figure 38).

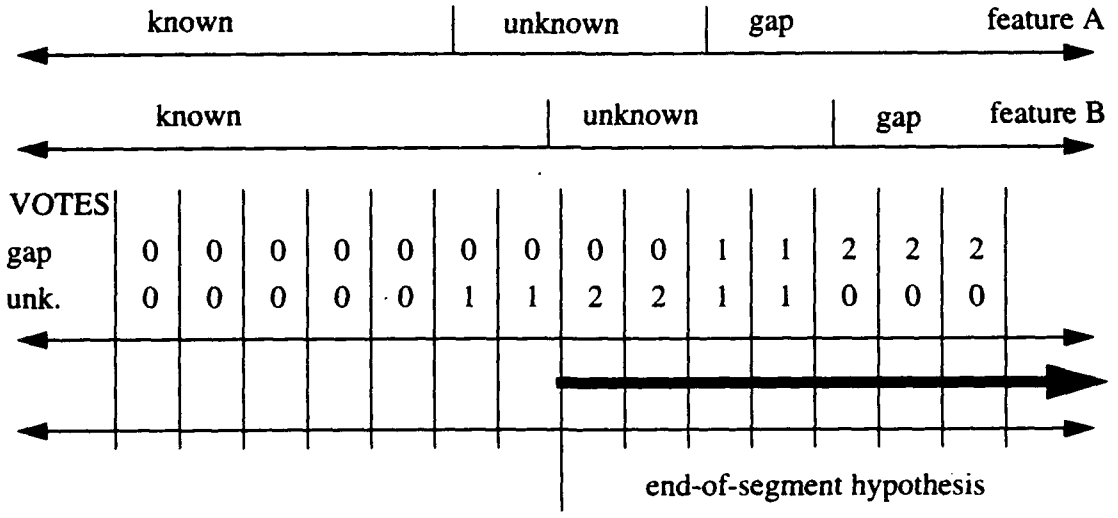



**Figure 38: Interval voting for end-of-segment hypotheses**

An example of this process on a real road is shown in Figure 39. YARF is tracking the double yellow line on the left of the lane and the solid white line on the right of the lane. The white dots running down the middle of the image show YARF's estimate of the center of the lane. The map indicates that both the features being tracked end at the next intersection. The trackers failed to find the features in the top three windows for both features. These failures generate gap intervals in the descriptions of the individual features. The voting scheme detects the joint absence of the two features, and here has marked the hypothesized end of road segment with a horizontal white line in the image.

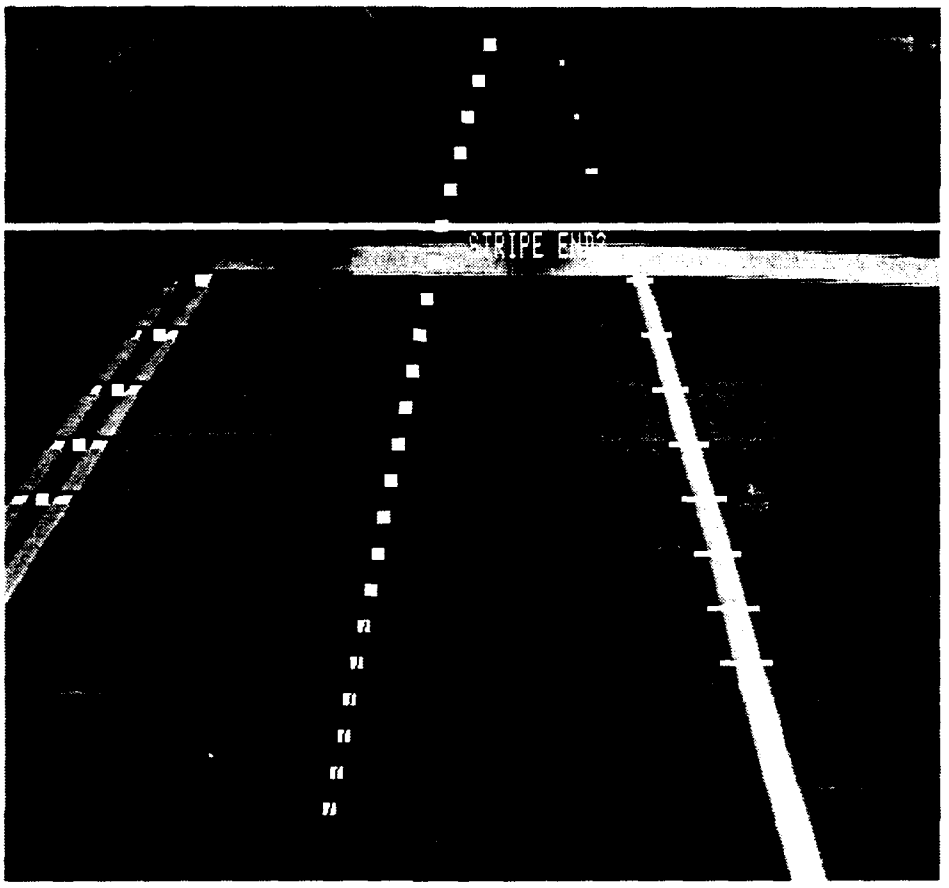



**Figure 39: Example of end of road segment detection**

## 5.4 Verification of end of stripe hypotheses

Once YARF has an estimate of the distance to the end of the current road segment, it can then combine that information with the local intersection geometry stored in the map to predict the location of the next segment to follow. As can be seen in Figure 39, intersections of city streets generally cover too much area to be seen in the field of view of a single camera. YARF currently uses a second fixed camera to provide additional coverage, and is therefore limited at the moment to making turns to one side. Adding additional fixed cameras to cover more area or use of an aimable camera would permit general turning. YARF applies 10 trackers at one meter intervals along the predicted location of the next road segment. If they succeed in finding the new road segment, then YARF begins to track the next segment and plan a path through the intersection. If verification fails to see the expected next segment, then the system assumes that the end of segment hypothesis was a false alarm and attempts to continue tracking the current segment.

Figure 40 shows an example of intersection detection and verification performed at an intersection near the Carnegie Mellon campus. The overhead view shows the vehicle and current road segment in the lower left, with the road ending at the estimated start of the intersection. The asterisks extending past the end of the center line of the current road

segment are the tracker failures which triggered the intersection hypothesis. In the upper right are the points detected along the center line of the intersecting road, with the road features for the next segment drawn in.

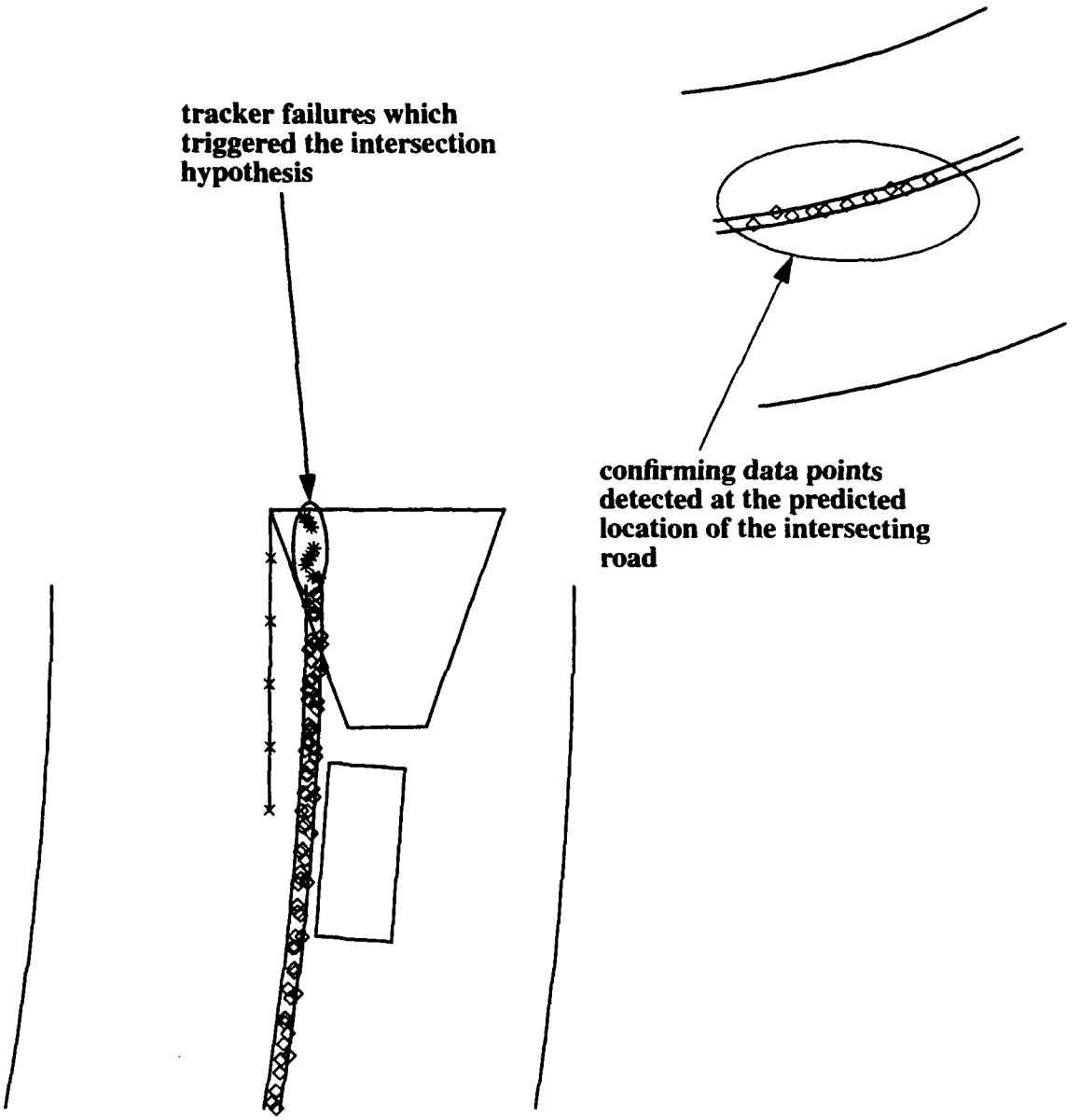



**Figure 40: Intersection verification**

## 5.5 Path planning for lane changes and intersection navigation

While YARF is tracking the center of a lane it uses a pure-pursuit routine built into the controllers of the NAVLAB I and NAVLAB II vehicles. Pure-pursuit is described in [56]. YARF uses a different path planning strategy to perform lane changes and navigate through intersections. Given the vehicle location and orientation at the start of the maneuver, and the desired vehicle location and orientation at the end of the maneuver, YARF generates a path

using a routine from the path planner developed by Stentz for his thesis [47]. This routine generates a path that consists of a turn of minimum radius, followed by a linear path segment, followed by a second turn of minimum radius. If there is no feasible path from the start point to the end point, the model of the destination lane shape is used to select a destination point 0.5 meters further along the destination lane. The path planner is then invoked again to see if there is now a feasible path. This process of moving the destination further out and testing for a feasible path continues until one is found. YARF then generates a set of waypoints which are sent to a path tracker. The waypoints are regenerated in each image cycle in order to accommodate errors in the location of the destination lane due to deviations of the terrain from the flat earth model.

Figure 41 shows an example of the path generated to perform a lane change in a simulated vehicle run. Figure 42 shows an example of the path generated to turn left through an intersection in a simulated run. This path planning method does not consider obstacles or traffic control signs and signals, and would need to be modified appropriately in order to use it in closed-loop runs on city streets. Live intersection detection and verification runs have been performed in open-loop for safety reasons.

## 5.6 Directions for future work

While gap extraction and intersection hypothesizing have been tested in many live runs, only a small number of live experiments have been performed to test the intersection verification methods. Additional experimental work is needed to test the robustness of the approach used.

Currently gaps in broken lines are not treated any differently than gaps in solid lines. The system should use the additional context provided by the knowledge that a line is broken rather than solid to help avoid possible false alarms due to spurious gaps in other features.

In general, the gap extraction process is primitive. It does not detect and eliminate failures due to transient events such as passing cars. While requiring multiple failure support for gaps eliminates most problems due to false failures, false feature detections can also pose a problem, although a much more rare one. An isolated false detection will affect the estimate of the end of the current stripe, and thus will also affect the prediction of the location of the new road segment. This may result in an incorrect failure to recognize an intersection. An optimization approach combining the failure information from all features would be superior.

The local map stores data points sorted by $y$ value in the current camera-based ground frame. As a result, the implementation of the interval voting parameterizes the bins it uses by distance in front of the camera. If the $y$ axis of the local map is not parallel to the lane tangent at the intersection then different features will end at different distances in front of the vehicle. This happens due to camera pan, road curvature, or detection of the intersection while the vehicle is in the process of changing lanes. The result is error in the system's estimate of the location of the end of the road segment. It would be better to parameterize the bins by arc length along the spine of the road segment and correct the feature interval endpoints accordingly.

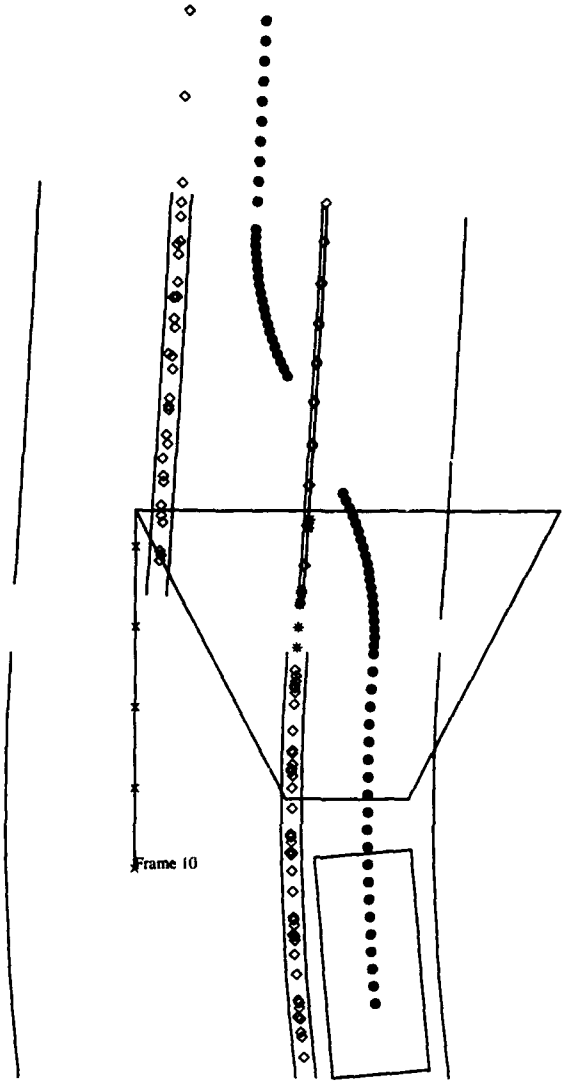



**Figure 41: Path planning for a lane change**

The techniques described allow YARF to handle changes in lane structure and intersections which are predicted in the road map used by the system. Situations may occur in which the road features change in ways which are not predicted by the map. This could be due to construction, or widening of the road, or changes in the marked lane structure since the time the map was constructed. Currently YARF returns control to the human safety driver if all the features being tracked vanish. A fast implementation of the SHIVA algorithm described in Chapter 6 could be used to test the hypothesis that the map's model of the current segment is not correct and to generate an updated segment model.

Similarly, the need for the geometry of the intersections in the map is also limiting. While computerized driver's aides for route planning can provide information about the number of roads meeting at an intersection and their rough relative orientations, they are unlikely to have detailed knowledge of the relative location of lane markings on the roads. Work is

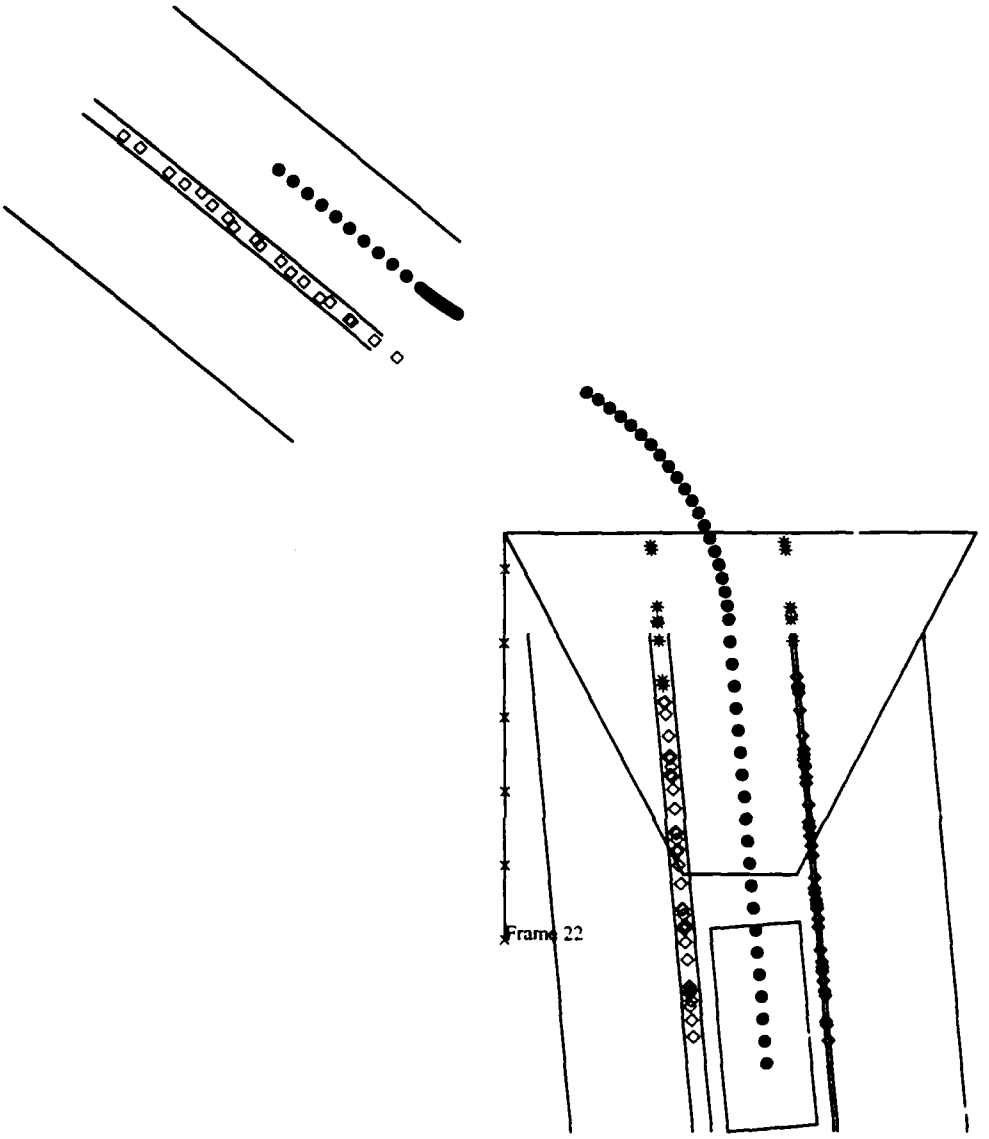



**Figure 42: Path planning to turn through an intersection**

needed in the domain of automatically extracting the branching roads at an intersection. This will involve active vision to move a camera to cover the full area of typical city intersections.

In the longer term, the flexibility of the system could be extended by the addition of other types of landmarks into the system. A road segment might have a distinctive sign on the side of the road shortly before the next intersection. Detection of the sign could be used to avoid early false intersection hypotheses. This leads into all of the issues of feature and sensor selection raised in [4]. Such extensions are necessary to perform navigation given the kind of models provided in typical directions generated by humans, as was pointed out in Section 5.1.

# 6 Initial Detection of Road Features: the SHIVA Algorithm

## 6.1 Road model initialization

Given an initial estimate of vehicle location and road curvature, the YARF system is able to follow a road segment which has a constant lane structure. Constraints provided by the system's model of road geometry and appearance enable it to respond to expected changes in lane structure and the approach of intersections. The methods which YARF uses to perform these tasks have been addressed in the preceding chapters. This chapter discusses the problem of performing the initial location of the road and recovering from unexpected changes in the lane structure.

Early versions of YARF required operator intervention to derive their initial estimate of the road location. This could be done in two ways. In the first, the user specified a feature from a preexisting road model and used a cursor to select three points along the left edge of that feature. In the second, the user selected points on the left and right edge of the lane the vehicle was in, resulting in construction of a lane model and the initial estimate of road position. Both are clumsy solutions from the perspective of automating the road following task. They require use of a cursor to pick points on the road, an awkward method of initialization compared to the convenience of pressing the start button on a cruise control. A more desirable solution is for the system to extract the road structure in an initial image and either match it to a preexisting feature cross section model, or build the model from scratch.

Similarly, in cases where YARF is unable to locate the expected road features the system has no other choice than to stop sending commands to the vehicle controller, inform the user that the road has been lost, and exit. Again, a more desirable solution is for the system to examine the current image and extract a model of the visible road structure in order to recover and update its model of the road. The ability to extract the road lane structure based on image data is also an important step towards the level of autonomy needed for integration of road following with other driver's aids. Systems such as Travelpilot [7] will provide maps of the road network which give roughly accurate geometry at intersections, but are unlikely to include details about lane structure and do not give exact position data. This will require road following systems to extract the local lane structure for themselves.

The algorithm which does this should extract as much of the lane s: ucture of the road as possible rather than simply detecting pavement edges or boundaries of the current lane. It should use an appropriate generic road model rather than require training by the user. The SHIVA algorithm (*Sobel/Hough Identification of VAnishing points*) has been developed to meet these requirements. The domain model used considers roads to be composed of features with constant separation from each other. Within a short horizontal band of the image the road is approximately straight. The features which compose it are approximately linear and parallel on the ground. As a result, the perspective projection of the features into the image plane should result in their meeting at a shared vanishing point on the horizon. Sobel edge detection is used to extract edge points in the image, and Hough transform techniques are used to determine the road vanishing point and filter out spurious edges.

SHIVA embodies the same philosophy used in the basic lane following loop: use model constraints to filter the results of noisy segmentation techniques. It embodies this philosophy in a bottom-up manner rather than in a top-down manner.

## 6.2 Related work

SCARF [9] uses a Hough transform to filter the results of a noisy color classification. Road and nonroad color classes are modeled by multivariate Gaussian clusters in RGB space. Each pixel is classified as either road or nonroad. Each road pixel votes in a Hough space for every road it could lie on. Each nonroad pixel votes against every road it could lie on. Even though the color classification used is not perfect, the global voting results in robust location of the road. Similarly, in the UNSCARF system [9] regions are found by an unsupervised color classification segmentation. A search is then done to locate the set of regions most consistent with a given road model. Both these systems locate only the road surface, ignoring any internal markings such as lane boundaries.

Polk and Jain [39], extending earlier work by Liou and Jain [33], developed an algorithm which is boundary based rather than region based. In the case of a straight road the road features have parallel boundaries which project into the image as lines meeting at a common vanishing point. In the case of curved roads this is true as a local approximation, i.e. within some range of rows the feature boundaries are close to linear and converge at a shared vanishing point. Their algorithm uses the Sobel edge detector to find boundary points in the image. They look only for the left and right edges of the current lane rather than trying to extract all the visible road features. They also do not attempt to classify the features by type.

The MARF system developed at the University of Maryland [57] has a bootstrap mode to perform the initial location of the road edges and initialize a predict-verify feed-forward mode. Sobel edge extraction is performed to locate high contrast points. A histogram of pixel gradient direction weighted by the pixel gradient magnitudes is computed to find dominant edge directions in the image. For each peak in the direction histogram, edge pixels with a gradient direction close to the peak direction vote for a line intercept. Intercepts with sufficient support are reported as features.

Suzuki, et al. [48] track the left and right lane markers of the current lane using an adaptive thresholding scheme and Hough line detection. Several rows are sampled around the previous location of the left and right lane markings, and the average and maximum pixel value computed for each of these rows. Thresholds are set for the left and right markers based on a weighted average of the average and maximum intensities for the sampled rows. The pixels in the area around the location of the markers in the previous image are thresholded to extract the points corresponding to the lane markers. A Hough transform is used to compute the parameters of the lane edges.

## 6.3 Description of the SHIVA algorithm

The SHIVA algorithm improves on previous techniques in two ways. First, it extracts the full visible lane structure of the road rather than just the pavement edges or the edges of the lane the vehicle is in. Second, it applies the YARF feature trackers to attempt to identify the type of each feature. The previous techniques described above identify road or lane

geometry, but not the nature of the features. Identifying the left edge of the current lane as a yellow line rather than a white line is important for making tactical level driving decisions. The SHIVA algorithm involves the following stages of processing:

- Edge pixel extraction. This involves substages of

  Preprocessing the color input image to produce a single band image to use as input to the edge operator.

  Applying the Sobel edge operator to the preprocessed image.

  Eliminating nonmaxima and thresholding the Sobel gradient magnitudes.

- Division of the image into a small number of horizontal sections. Within each section SHIVA performs the following steps:

  Each edge point votes for the lines it could lie on.

  Peaks in the Hough accumulator array are located, corresponding to linear features.

  The linear features identified in the previous step vote for vanishing points on the horizon. The vanishing point with the strongest support is assumed to be the vanishing point of the road within that section. The features voting for that vanishing point are assumed to be road features.

- Connection of features across section boundaries.

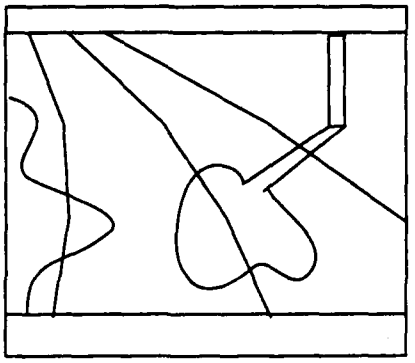
- Classification of extracted features using tracker responses and spatial constraints.

These stages are illustrated in Figure 43 and described in detail below.

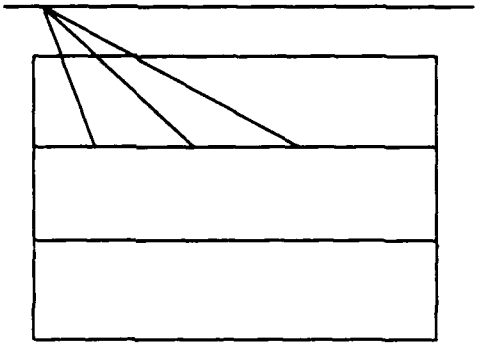
### 6.3.1 Line parameterization for the SHIVA algorithm

SHIVA assumes that the horizon appears horizontal in the image plane, although it need not be in the camera's field of view. A linear feature in the image is parameterized by the column at which the line it lies on intersects the horizon row and the angle it makes with the horizon row. Figure 44 shows the relationship between the coordinates of a point lying on a line in the image and the parameters defining that line. The relationship between the coordinates of a point in the image and the parameters of a line passing through it is defined by the equation $\tan(theta) = (r - horizon\_row) / (c - vanishing\_column)$.

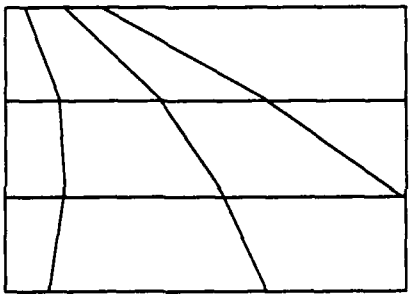
This parameterization was chosen for two reasons. First, it makes identifying linear features which share a common vanishing point very easy. Second, using orientation rather than slope as a parameter allows even partitioning of line orientations by equally spaced bins in the parameter space.



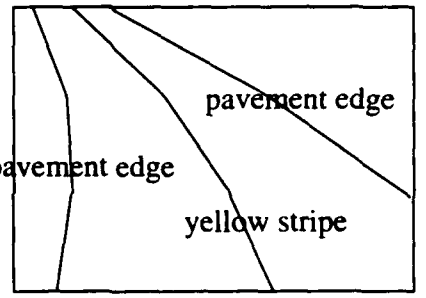
Stage 1: edge pixel extraction



Stage 2: division of the image into horizontal sections; edge pixels vote within each section for linear features; features vote within each section for shared vanishing points



Stage 3: connection of features across section boundaries




Stage 4: Classification and matching of extracted features to map model

**Figure 43: Summary of stages in the SHIVA algorithm**

### 6.3.2 Preprocessing and edge point extraction

The color image produced by the camera is subsampled to produce a 120 by 128 image from the original 480 by 512 image. SHIVA allows the user to select from a number of options for the color feature which is used in edge point extraction. Currently supported choices are:

intensity image (average of RGB);

raw red, green, or blue band;

normalized red, green, or blue band; or

(blue - red) image.

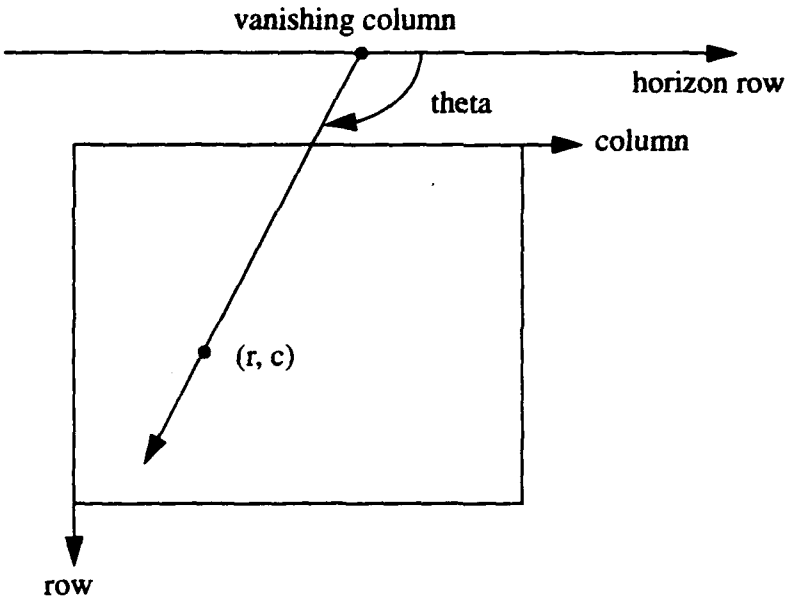



**Figure 44: Relationship between coordinates of a point and the parameters of a line passing through the point.**

Experiments to date suggest that the raw green band works well, giving good contrast between pavement and stripes as well as pavement and nonpavement.

Any segmentation technique could be used which returned boundary points in the image with an estimate of the boundary orientation at each point. The Sobel edge operator (described in [2]) is used due to its speed and simplicity. It computes estimates of the $x$ and $y$ gradients in the image ($gx$ and $gy$) by cross correlation of each 3 by 3 window in the image with the masks shown in Figure 45. The gradient magnitude at each pixel $w_{0,0}$ is computed as $magnitude = \sqrt{gx^2 + gy^2}$. The gradient orientation is equal to atan $(gy/gx)$. The edge orientation at the pixel is computed as $theta = 1.5\pi - \text{atan}(gy/gx)$ After theta is corrected to lie between 0 and $2\pi$, if $theta > \pi$ then $theta = theta - \pi$. This makes the theta value consistent with the way in which line orientation is parameterized and ignores the difference of $\pi$ in edge orientation between lines with the same orientation and opposite contrast.

$$\begin{bmatrix} w_{-1,-1} & w_{-1,0} & w_{-1,1} \\ w_{0,-1} & w_{0,0} & w_{0,1} \\ w_{1,-1} & w_{1,0} & w_{1,1} \end{bmatrix} \otimes \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = gx \qquad \begin{bmatrix} w_{-1,-1} & w_{-1,0} & w_{-1,1} \\ w_{0,-1} & w_{0,0} & w_{0,1} \\ w_{1,-1} & w_{1,0} & w_{1,1} \end{bmatrix} \otimes \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = gy$$

**Figure 45: Masks used to compute the $x$ and $y$ gradients for the Sobel edge detector**

The edges are thinned by a very simple nonmaxima suppression technique which looks at the orientation of the gradient of a pixel and the magnitudes of the gradients of its 8-neighbors. Gradient orientations are partitioned into the four areas A, B, C, and D shown in Figure 46. The central pixel in a 3 by 3 neighborhood is compared to its neighbors along either the center row, column, or diagonals, depending on orientation. If it is not greater than or equal to the magnitude of both of the neighbor pixels along the gradient direction, then the gradient magnitude for that pixel is set to zero.

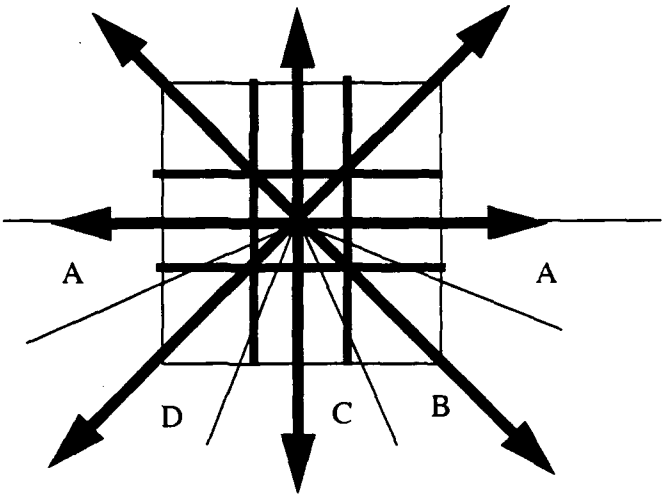



**Figure 46: Partitioning of gradient orientations for nonmaxima suppression**

Next the gradient magnitude image is thresholded to extract potential edge pixels. There are two methods provided for specifying the gradient threshold. The first is a fixed gradient magnitude. A value of 15 intensity levels (where the image intensity varies from 0 to 255) works well on the test images examined to date. The second method is percentile based. The user specifies a percentile cutoff on the gradient magnitude, and histogramming is used to determine the appropriate threshold.

### 6.3.3 Voting for line segments and shared vanishing points

The candidate edge points now vote for possible lines in the line parameter space. The image is partitioned into a small number of horizontal sections in order to accommodate the change in road vanishing point due to curvature in the road. Edge points in each section vote separately for linear segments and the overall vanishing point for that section.

While bins are provided in the parameter space for line segments whose vanishing points lie off the image by some fraction of the image width, in the case of roads with high curvature the road features will have vanishing points far outside the image. As a result, the top of the image is cropped. Also, only one lane or road edge is typically visible near the bottom of the image. Since multiple road features are not visible in this area to vote for a shared vanishing point, the bottom of the image is also cropped. The limits of this cropping depend on the camera field of view and tilt angle.

Each potential edge point located by the Sobel operator votes for bins in the parameter space. The bins correspond to lines through that point and a particular choice of vanishing column. The votes are not weighed by the edge gradient magnitude, as features differ in their relative contrast, and shadows may have higher contrast than the features. Edge pixels only vote for lines whose orientation is within a specified tolerance of the estimate of gradient orientation for that pixel. This reduces the effect of noise and shadow edge pixels. In addition, the pixel votes are weighted by the difference between the Sobel estimate of the edge orientation and the bin's line orientation according to the formula $vote = 1 - (\frac{min(angle\_diff, \pi - angle\_diff)}{\pi/2})$, where $angle\_diff = |edge\_angle + line\_angle|$. This weighting reduces the influence of spurious edge points.

Candidate line segments now have to be extracted by peak detection in the accumulator array. In order to avoid underweighting segments which are clipped by the boundaries of the image, the votes for each bin are normalized by the visible length of the segment corresponding to that set of parameters. Bins representing very short segments (length < 2 pixels) have their votes set to zero, as this normalization would otherwise overweight such very short segments.

The normalized accumulator space is now thresholded. The bins which remain represent candidate segments which have edge point support for more than some fraction of their visible length. The votes for all orientations are summed for each possible vanishing column. The vanishing column which has the most total support is judged to be the road vanishing column for the current section of the image, and the above threshold bins in the accumulator array which correspond to lines with that vanishing column are the candidate road features for the current section of the image. This process is repeated for each of the horizontal sections of the image separately.

### 6.3.4 Connecting features across section boundaries

At this point in the SHIVA algorithm the features are still represented implicitly by the accumulator arrays for the horizontal sections. The next stage of processing converts that information into a symbolic description of the features detected by the Hough voting procedure. Each feature is represented by a sequence of (*row, col*) pairs which represent the intersection of the feature with the top row of each section.

Knowing the vanishing point corresponding to each horizontal section of the image permits the extrapolation of the location of a feature from one section through the other sections. Given the vanishing column vc[N] and orientation theta[N] of a feature in section N, and given the vanishing column in section N+1 is vc[N+1], the orientation of the feature in section N+1 is constrained to be $theta[N+1] = \text{atan}(\frac{(c[N+1] - vc[N])}{(c[N+1] - vc[N+1])} \tan(theta[N]))$ (see Figure 47). Similarly, we can constrain the orientation of section N-1. In this way, given a bin corresponding to a feature in one section, the bins corresponding to the feature in the other sections can be determined.

All bins in the accumulator arrays are initially marked as unused by any symbolic feature. The accumulator arrays for the image sections are scanned from the top section downward. When a bin is found which corresponds to a feature voting for the vanishing point in that section, and which has not been marked as used already, a new symbolic feature description

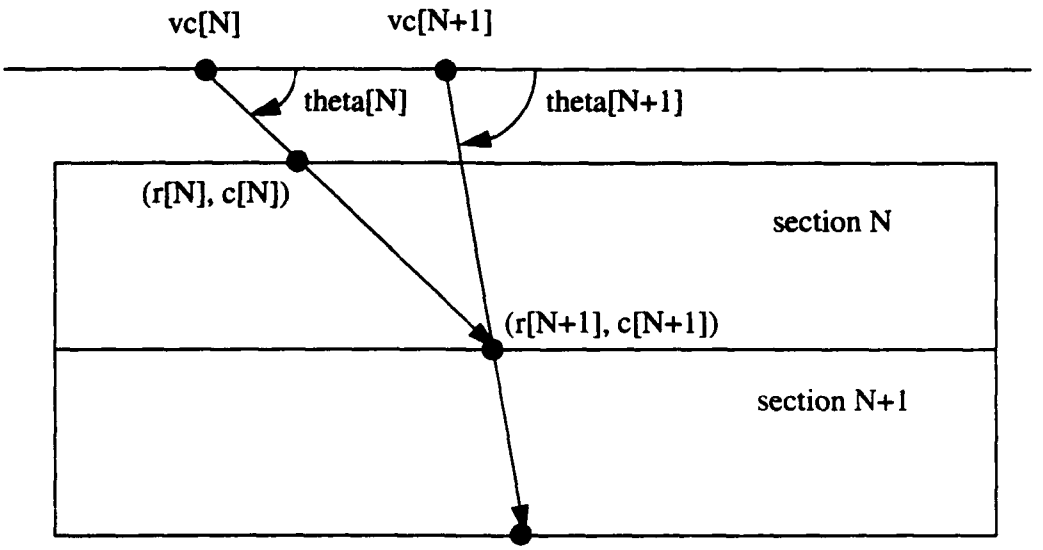



**Figure 47: Geometry relating the parameters of a feature which extends across a section boundary**

is generated. The sequence of vanishing points is used to determine the location of that feature lower in the image, and bins which correspond to the feature in lower sections are marked as used. After the accumulator has been scanned and all the features located, the positions of the features in sections where they were not located are determined using the constraints provided by the sequence of vanishing points.

### 6.3.5 Classifying features using the trackers

At this stage in processing SHIVA has extracted the geometric description of the features, but has not associated any semantics with them. The tracker algorithms used in the main road following loop are used to classify the features into yellow and white painted stripes, road shoulders, and "other" (shadow edges, etc.). Each feature has the yellow hue and oriented bar trackers applied at five points along the visible length of the feature in each of the horizontal image partitions. This creates four score values. The first is the fraction of the oriented bar windows which returned a positive response. The second is the fraction of the yellow hue windows which detected yellow stripe pixels to the right of the feature. This score is called left_yellow because a high value suggests that the feature is the left edge of a yellow stripe. The third is the fraction of the yellow hue windows which detected yellow stripe pixels to the left of the feature. This score is called right_yellow because a high value suggests that the feature is the right edge of a yellow stripe. The fourth is total_yellow, the sum of left_yellow and right_yellow.

One simple spatial constraint is used in the feature classification: whether the feature intersects the bottom of the image to the left or right of the center column. The vehicle cannot be to the left of any yellow markings on the road, and the vehicle cannot be to the right of the right shoulder of the road. This constraint is combined with the four score values described above in a decision tree, which is shown in Figure 48. Currently SHIVA does not use any higher level constraints involving multiple features. Also, the decision tree does not include broken white stripes. These extensions have been left for future work.

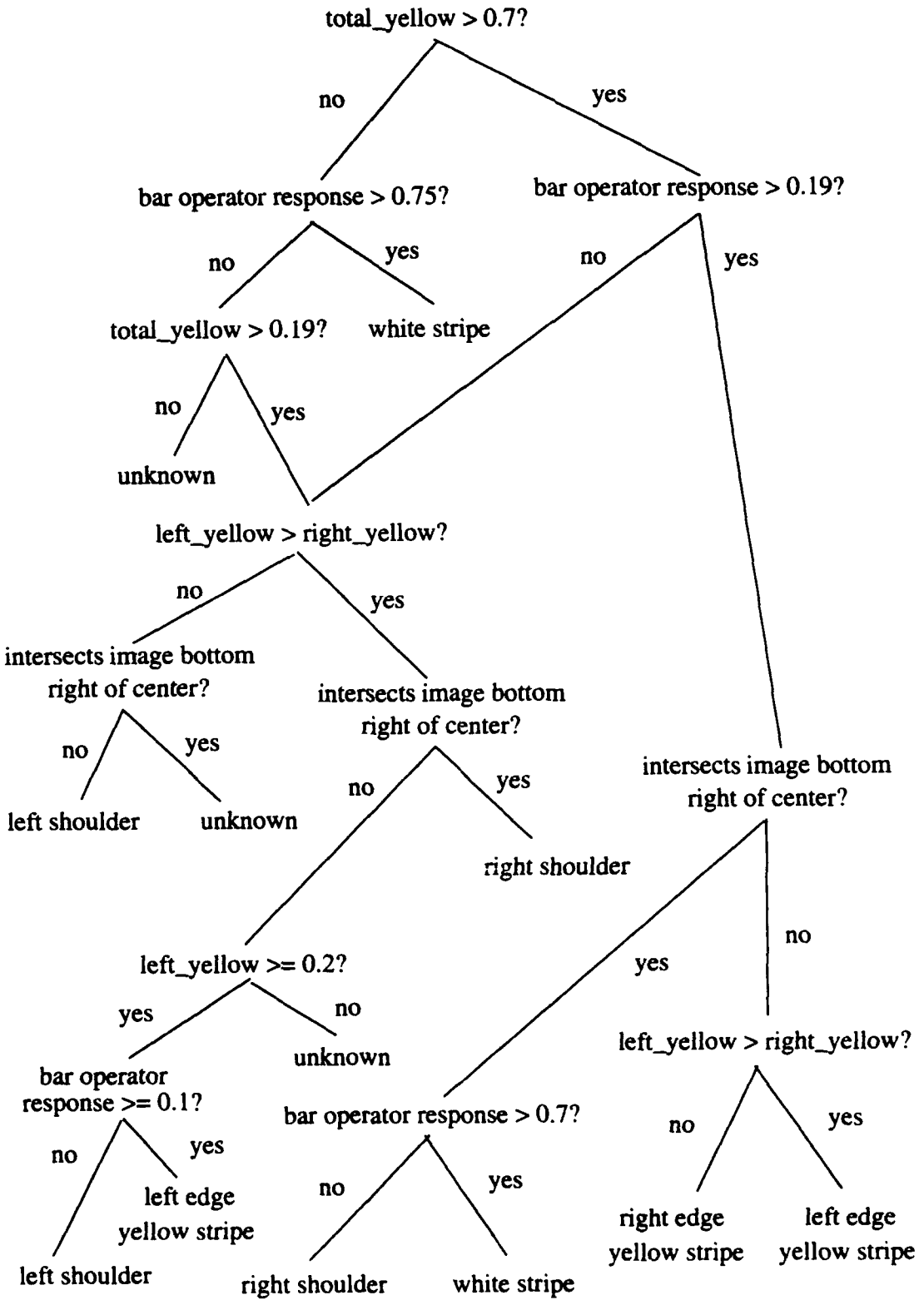



**Figure 48: Decision tree used to classify features in SHIVA**

## 6.4 Results

Figure 49 shows two examples of applying the SHIVA algorithm to images with strong shadow edges. The original images are shown on the left, the processed images on the right. The top and bottom portion of the processed images are the grey scale values in the areas cropped by the algorithm. The middle portions of the processed images show the detected edge pixels in light grey and the extracted features in white. The numbers next to the extracted features are the identifiers used by SHIVA to refer to the features. In both images the white stripe on the right passes through noisy shadow edges, but the shared vanishing point constraint allows SHIVA to successfully locate the feature position. Edge tracking algorithms would become confused by the textured shadow edges, which would offer multiple possible continuations.

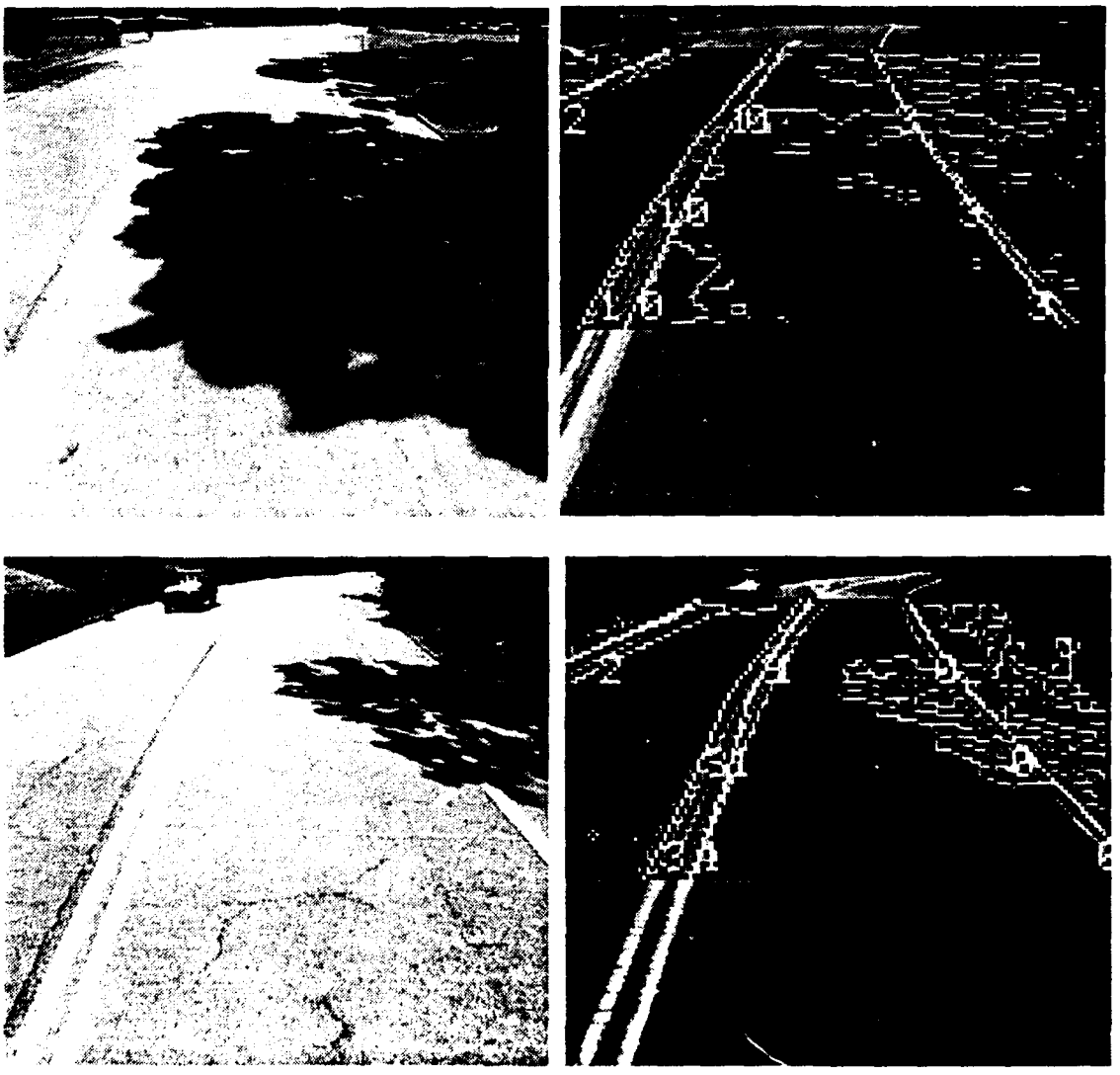


**Figure 49: SHIVA examples with features going through textured shadow edges**

Figure 50 shows the result of applying SHIVA to images in which the road curves. In the top example, SHIVA locates both sides of the double yellow line, the white stripe on the right edge of the lane, and the shoulder edge on the right. In the bottom example, SHIVA locates one edge of the double yellow line, the white stripe on the right side of the lane, and the bottom of the guard rail to the right of the road. Notice that the feature locations are not totally accurate due to the discrepancy between the features and their linear approximations, but are well within the tolerance necessary to use them to make initial predictions for the feature trackers.

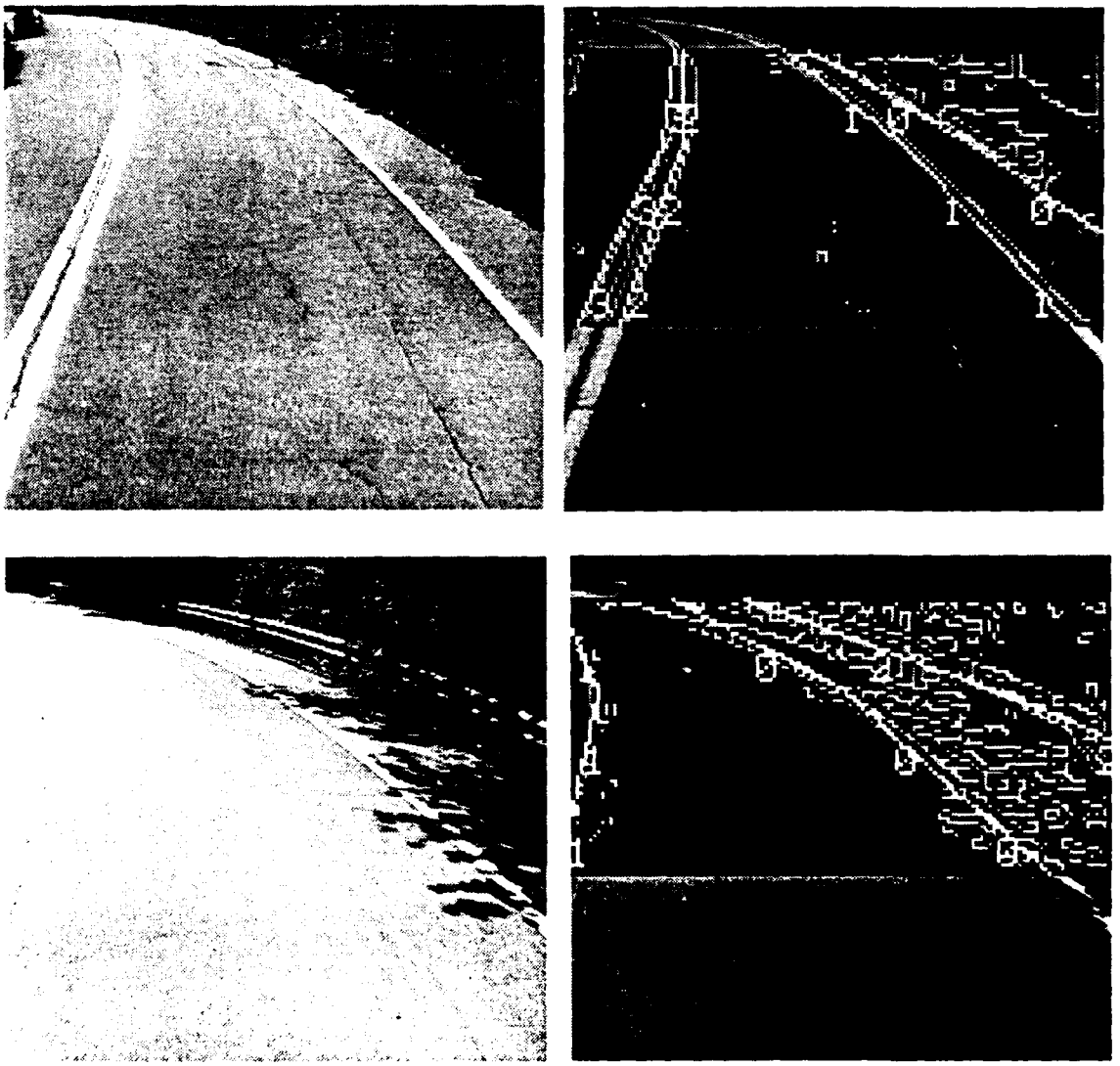


**Figure 50: SHIVA examples with curved roads**

The images in the preceding examples were scenes of a road with solid markings. Figure 51 shows the results produced by SHIVA for input images containing broken painted stripes. The top pair of images show the input image and SHIVA result for a scene taken on a local divided highway. SHIVA successfully locates the solid white stripe on the right side of the lane (feature 3), the broken white stripe on the left side of the lane (feature 5), the bottom

edge of the concrete barrier on the left side of the road (feature 4), and three of the edges of the concrete barrier on the right side of the road (features 0, 1, and 2). The bottom images show the SHIVA result for another location on the same road.

Figure 52 shows the results of the SHIVA algorithm on three images taken on a multilane road near campus. The right edge of the vehicle's lane is a broken white stripe. The shared vanishing point constraint allows the extrapolation of the location of the lane edge in the section of the image where it is missing.

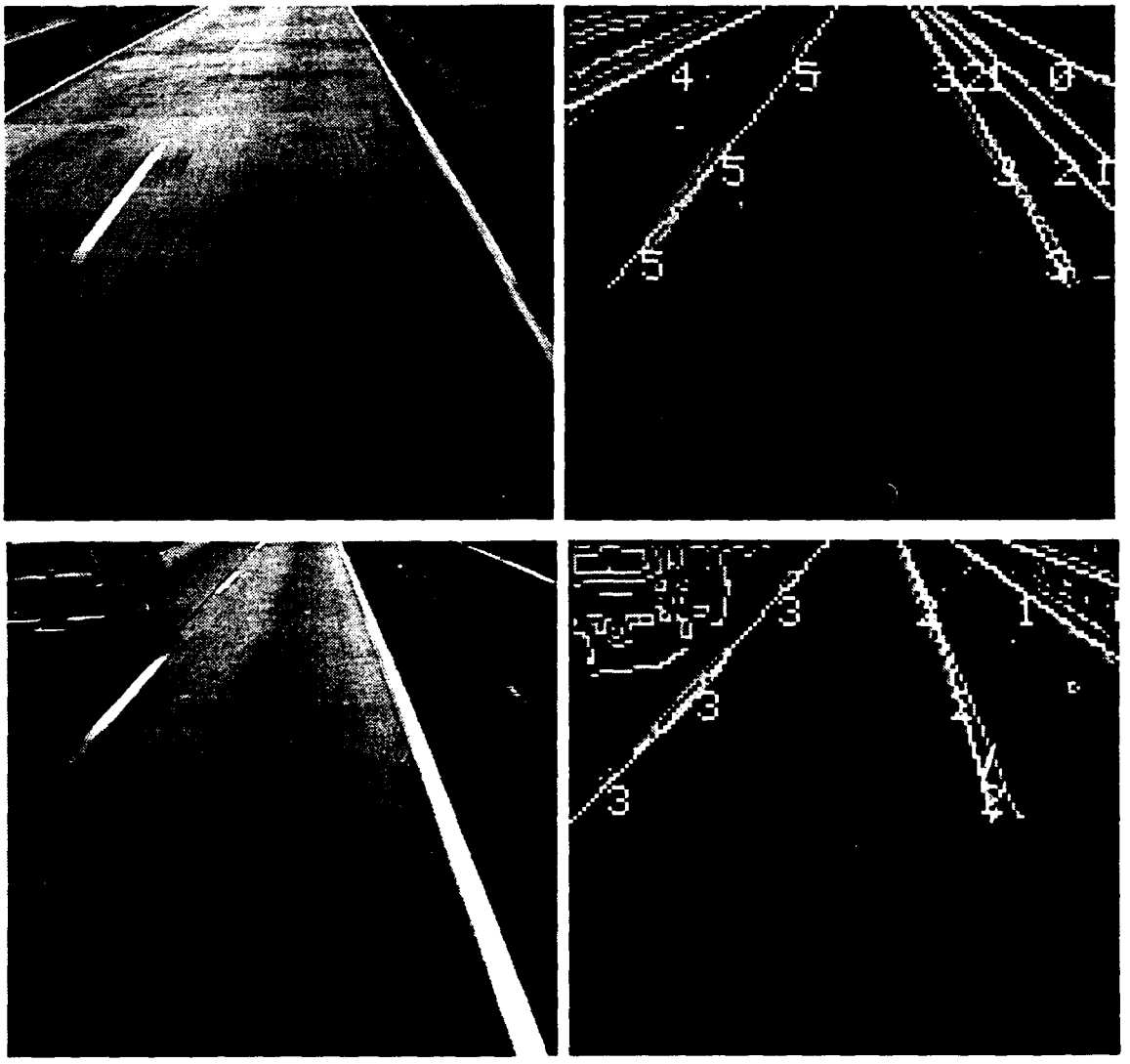


**Figure 51: SHIVA examples, road images with a broken white stripe and concrete barrier**

The above results demonstrate the ability of SHIVA to extract the geometric information about road features in the image. A separate experiment was performed to test the feature classification decision tree described above. The decision tree for feature classification was derived by examining the feature scores in a set of 17 images. The resulting hand-

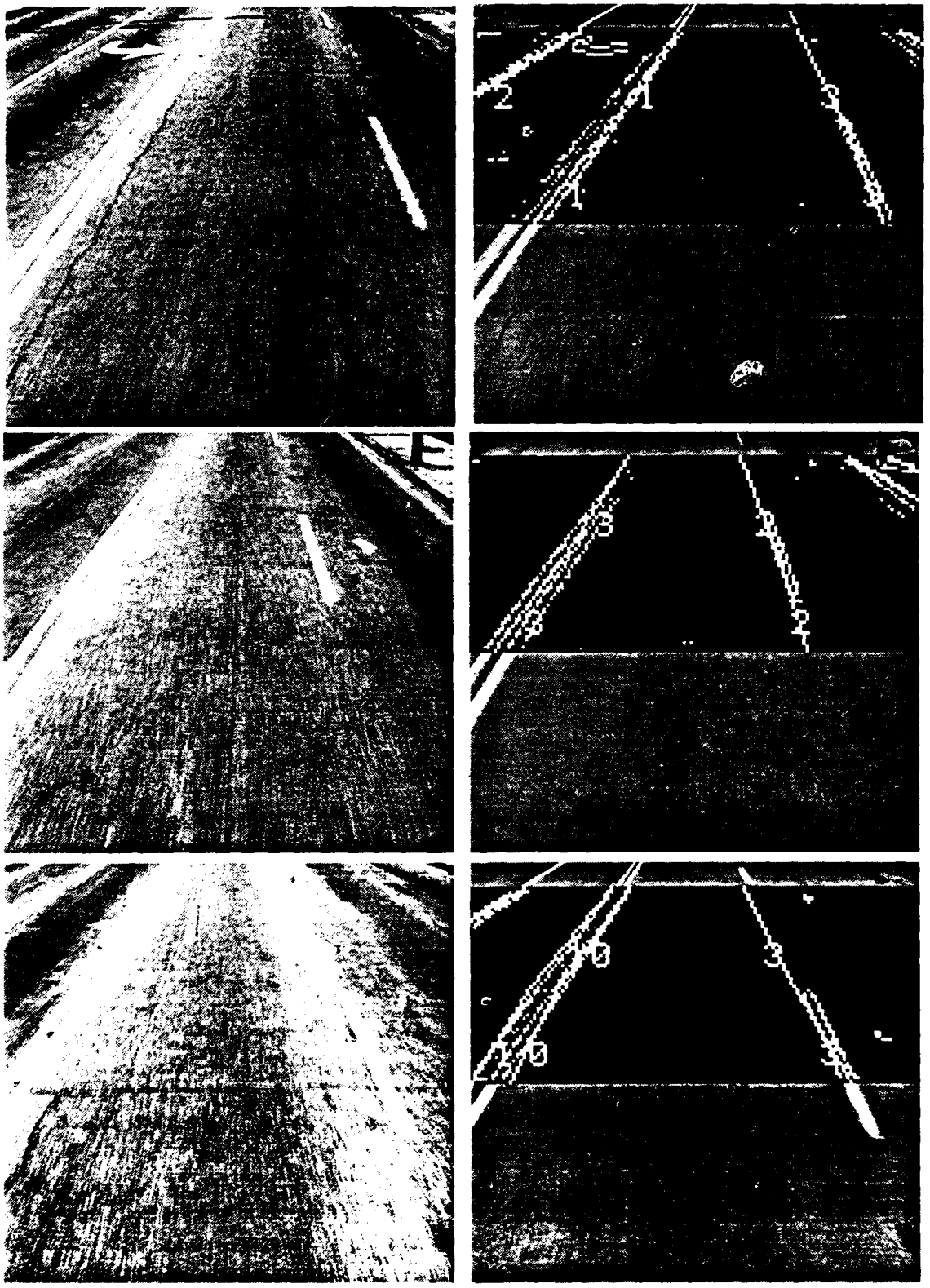


**Figure 52: SHIVA examples, street with a broken white stripe**

constructed tree was then tested on a set of 30 additional images. All 47 images came from a data set taken on Schenley Drive, a road near campus. Schenley Drive is a two lane road, with a double yellow stripe down the center and white stripes marking the outer lane edges. Portions of the road have a guardrail or fence running alongside. SHIVA found 156 features in the 30 test images, 125 of which were correctly classified. Of the 31 errors, 16 were false features due to shadow edges and edges from a fence and guardrail which run along sections of the road. Seven were due to the oriented bar tracker not being able to detect the white stripe on the left edge of the left lane. Due to the limited camera field of view, that stripe was visible only in the distance. As a result, that stripe appeared very narrow in the images. The confusion matrix for the results is shown in Figure 53. While these results are based on a relatively small data set, they confirm the basic feasibility of using the YARF trackers to classify features detected by SHIVA.

| actual / classification | yellow stripe | | shoulder | | white stripe | | |
|---|---|---|---|---|---|---|---|
| | left edge | right edge | left | right | left lane | right lane | other |
| left edge, yellow stripe | 23 | 0 | 0 | 0 | 0 | 0 | 0 |
| right edge, yellow stripe | 0 | 25 | 0 | 0 | 0 | 0 | 0 |
| left shoulder | 0 | 1 | 16 | 0 | 0 | 1 | 1 |
| right shoulder | 0 | 0 | 0 | 21 | 0 | 1 | 11 |
| white stripe | 0 | 0 | 0 | 1 | 1 | 27 | 4 |
| unknown | 0 | 0 | 2 | 1 | 7 | 1 | 12 |

**Figure 53: Confusion matrix for SHIVA feature classification experiment**

## 6.5 Evaluation of SHIVA

### 6.5.1 Use of the Sobel edge operator to detect feature boundaries

The SHIVA algorithm uses the Sobel edge operator because it is fast, and it provides an estimate of edge orientation. The disadvantage of using the Sobel operator is the difficulty of selecting a gradient threshold. The image on the right in Figure 50 is a good illustration of this problem. The edges of the double yellow line have lower contrast than the strong

shadow and texture edges on the shoulder of the road. As a consequence, a percentile based thresholding scheme will miss the yellow stripe edges in such a situation. A fixed gradient magnitude threshold appears to be the better choice.

The SHIVA algorithm is not dependent on the use of the Sobel operator to extract boundary points, and other segmentation techniques could be substituted which were not as sensitive to texture or contrast. The main requirements for the segmentation technique are that it run in reasonable time and that it provide an estimate of boundary orientation. The first requirement arises from the need for the initial road location to occur rapidly enough for use in a system operating at traffic speeds. The second requirement arises from the need to constrain the portion of the Hough accumulator which each boundary point votes for in order to reduce the number of false peaks due to accidental alignments of noise edge points.

### 6.5.2 Handling road curvature by partitioning the image

SHIVA partitions the image in order to deal with the change in vanishing point caused by curvature in the road. The approximation of the road features as linear within each image partition allows the use of a two dimensional Hough accumulator array. This speeds up the voting stage and simplifies peak detection. In the case of broken features, however, the stripe may straddle a partition boundary so that there isn't sufficient support in either section, resulting in a failure to detect the feature. An alternative would be a parameterization of the road features which had the property that concentric arcs on the ground plane were easy to detect given their projections into the image. One possible choice has been identified, but not tested.

Placing the origin of the image coordinates at the center of the image, and placing the camera level so that the horizon falls along the center row, simplifies the camera calibration equations to $row = camera\_height / (y \times rf)$ and $column = x / (y \times cf)$, where $rf = pixel\_height / focal\_length$ and $cf = pixel\_width / focal\_length$ as defined in Section 4.2. Using the parabolic approximation to the road shape model as described in Chapter 4, $x = 0.5 \times k \times y^2 + head \times y + spine\_offset + feature\_offset$, the equation of the projection of one of the road features into the image plane is

$$column = \frac{head}{cf} + \frac{k \times camera\_height}{2 \times rf \times cf \times row} + \frac{(spine\_offset + feature\_offset) \times rf}{camera\_height \times cf} \times row$$

Thus, each feature in the image plane can be fit with a curve of the form $column = heading\_term + curvature\_term / row + offset\_term \times row$. All the features will have the same values for *heading_term* and *curvature_term*. Note that this implements the same model of road geometry as YARF, but in the image plane, with no need to backproject onto the ground plane.

### 6.5.3 Assumption of a flat ground plane

SHIVA assumes a single horizon row. If the terrain varies in slope, SHIVA may not be able to correctly locate all the road features due to their not having a vanishing point on the horizon row corresponding to the tangent plane at the vehicle location. The accumulator array contains all the information about the linear features within each horizontal image partition. One possibility would be to have pairs of lines in each section vote for their vanishing point, and cluster the votes to identify the best horizon row and vanishing point for that section. This would be similar to the "hill and dale" model of road geometry [36].

## 6.6 Conclusion

The SHIVA algorithm shows promise to provide a robust method for extracting and identifying the features composing the road in an image. The specialized trackers used for feature detection in the normal YARF predict-verify road tracking loop can be combined with simple spatial constraints to classify the features as stripes of different colors and road boundaries. Full integration of this algorithm with the YARF system would permit autonomous initialization of the system's estimate of vehicle location and road curvature. It would also permit recovery in cases where the road changed lane structure in a way not predicted by its map of the road network. Ultimately, it could lead to techniques which would allow the system to function without any prior knowledge of the lane structure of roads in the system's map of the road network.

# 7 Conclusion

This thesis describes YARF, a system for vision based road following. At the core of the YARF system is a model of the road structure which includes information on feature appearance and feature geometry. This model is central to every aspect of the system. It simplifies image segmentation by constraining feature location, orientation, and appearance. It generates the system of equations used to integrate individual measurements of feature locations into a single estimate of the road curvature and location relative to the vehicle. Finally, it creates expectations whose violation signals changes in road structure. YARF exploits the model characteristics to make innovations in each of the areas mentioned above.

The predict-verify-update loop used in YARF assumes that the model of road structure and appearance is provided by a map of the road network. For practical application it will be necessary for a system to be able to automatically generate the road model from image data. The SHIVA algorithm provides a mechanism for generating a road model by using constraints from a weak domain model to filter a noisy segmentation.

## 7.1 Importance of the model

The central theme of this thesis is the use of model information to simplify and increase the reliability of vision based road following. Three main areas were emphasized: segmentation, estimation, and reaction to changes in road structure.

One of the innovations of YARF is the use of the road model to select from a set of specialized segmentation techniques. Each type of feature is most reliably detected by exploiting a robust cue specific to that class of feature. Section 3.5 presented experimental results demonstrating the increased reliability achievable by using multiple trackers. In addition, using model information about feature location to constrain the area examined for a feature reduces the amount of computation required and the probability of false feature detections.

In addition, using the model to focus attention on the expected feature locations reduces the complexity of processing each image. The predict-verify-update loop used to track road segments typically looks at 30,000 or so of the pixels in a 512 by 480 image, and takes approximately half a second per image. SHIVA, which makes no assumptions about feature location, looks at all 15,360 pixels in a 128 by 120 image, and takes approximately 10 seconds per image.

The second area of emphasis was estimation. No segmentation technique is perfect, and any vision based navigation system will therefore find itself in situations where there are contaminants in the data. YARF demonstrates that the coherence constraints imposed by the scene model (in this case, a generalized stripe road model) can be combined with the least median of squares robust estimation technique to avoid the influence of such contaminants on the model fit. Without the coherence imposed by the model, this would not be possible.

The third area of emphasis was the use of model information to provide expectations about the manner in which the road will change appearance. By providing information about how all the features behave, brief gaps in individual features caused by passing cars or other transient problems can be ignored. This reduces the processing time spent on testing

incorrect hypotheses. By providing information about the road network which the system cannot currently extract autonomously, the map model of the road network enables YARF to execute more sophisticated missions than it could without the map.

## 7.2 Contributions

The YARF system makes a number of contributions to the field of vision based road following research. The first is the generalized stripe formalism for describing roads. This is the first system of road representation to incorporate information about road feature type and appearance into the road model. Previous systems included only geometric information about the road. In addition, the generic differential definition of a generalized stripe provides a common language for describing representations of road geometry.

YARF simplifies the image segmentation aspect of the navigation task by providing an architecture for combining model information about feature appearance with a library of specialized segmentation techniques. The interface between the feature trackers and the rest of the system is object oriented, identifying instances of feature trackers by their type and an index number. This hides knowledge of any state information used by the individual algorithms from the rest of the system, allowing easy expansion of the set of feature trackers. All of the trackers described in Chapter 3 were developed after the interface had been designed and implemented, and were easily integrated into the system.

YARF performs more robustly in the presence of contaminating data observations than existing systems based on a least squares approach. The use of the LMS robust estimation algorithm in combination with the geometric constraints provided by the road model allows the correct estimation of road curvature and vehicle location in situations where a least squares based approach would fail due to the presence of outliers in the data. Experiments using LMS for closed-loop runs in situations without outliers show that it performs well despite its lower efficiency compared to least squares. This opens up new lines of research into modifications to the LMS algorithm to handle imbalance in feature representation in the data set, and combinations of LMS with filtering.

YARF explicitly checks for and recognizes intersections and changes in lane structure. In previous systems, the disappearance of a lane edge would be ignored unless the system appeared to have totally lost track of the road. This could result in inappropriate behavior. YARF recognizes that such feature disappearances may signal the approach of an intersection or a change in lane structure, and uses map information to verify such changes.

Finally, SHIVA provides a robust algorithm for extracting the full lane structure visible in a single road image. It extends previous techniques by applying the YARF feature trackers to identify feature type as well as feature geometry. Such a technique is necessary in order to react to unexpected changes in road lane structure and initialize the system's estimate of road location and shape.

## 7.3 Comparison with other approaches

There are a variety of other vision based road following systems described in the literature. Each chapter has discussed relevant systems as they related to the functional subtask covered in that chapter: models of road geometry in Chapter 2, segmentation methods in

Chapter 3, model fitting methods in Chapter 4, map based navigation in Chapter 5, and road detection using a weak domain model in Chapter 6. It is useful, however, to also compare different approaches with respect to the issue of system architecture and integration. Vision based road following is not an end in itself, but a capability which can be used to help perform a variety of tasks. In order to evaluate the merits of a road following system it is necessary to consider the larger tactical navigation task which that system is intended to help perform. There are many such tasks: to stay in the current lane and maintain a fixed distance from the vehicle ahead; to navigate along a poorly defined, unmarked dirt road at a mine site while avoiding stationary obstacles; to drive in traffic through a network of city streets; and so on.

The YARF system is designed to operate in road environments where there are marked lanes. In the case of unmarked and unpaved roads, there are other approaches which offer greater robustness. The reason for this is the reliance of YARF on feature detection in small local search windows. Road edges are often poorly defined, and systems which exploit more global features such as SCARF [10] and ALVINN [40] can perform more reliably on such roads.

Within the domain of marked roads, the YARF system offers a number of advantages over other model based systems such as VaMoRs [13] and VITA [17]. It provides an extendable library of feature trackers applied in a model-driven manner; it incorporates robust estimation to avoid the influence of outliers on model parameter estimation; and it incorporates techniques for detecting and reacting to changes in road structure. More detailed quantitative comparisons with these systems are difficult, however. Both VaMoRs and VITA have driven longer continuous autonomous runs than YARF. Implementing YARF on a general purpose workstation provided the advantage of a convenient debugging and testing environment, but at the price of speed. YARF only runs at 2-4 Hz, making it impossible to try extended highway test runs.

The central idea in YARF is the importance of explicit models. At the other exteme is the ALVINN neural net road follower. The ALVINN system learns to drive along a given road after approximately five minutes of observing camera images and the steering behavior of a human driver. The network takes an image of the road as input, and produces a steering direction as output. The adaptive learning capability of the neural net architecture permits ALVINN to drive faster and on a broader range of road environments than YARF, but the implicit nature of the network's road representation generates two disadvantages for the ALVINN approach compared to YARF.

The first disadvantage relates to the need for human training of the ALVINN networks. Currently ALVINN needs a separate network for each different road it drives on. While the current YARF map includes the feature cross section for each road, the SHIVA algorithm provides a potential technique for extracting lane structure without human training or intervention. It is an open question whether a large class of roads can be followed using a relatively small closed set of neural networks. Research is under way to extend ALVINN by combining multiple nets in a manner which it is hoped will lead to generalization to previously unseen roads [22], but this is still work in progress.

The other disadvantage relates to the difficulty of extracting symbolic information about the tactical situation from an ALVINN network. ALVINN generates behaviors directly from its visual input, without constructing any explicit representation of the environment. As a result, symbolic information has to be inferred from the behavior chosen by the network. An

example of this is given in [51], where assumptions are made about ALVINN's driving behavior in order to infer the location of the road center. The representation of the road environment used in YARF is easily extendable to include the features used by the ULYSSES model of the tactical driving task [42]. ULYSSES provides a detailed, principled analysis of the tactical driving task for the case of urban driving with traffic. This model of the task can be discussed in terms of completeness and correctness. In order to use ALVINN to perform the same tactical task, it will be necessary to show either that the symbolic features used by a ULYSSES-like architecture can be inferred from the behavior of the neural networks in the system, or that the tactical task can be modeled equally well in a non-symbolic system.

## 7.4 Directions for future work.

In addition to the contributions described above, YARF provides an open ended framework for integrating different types of algorithms and models into an evolving navigation system with increasing competence. Such extensions can lie along a number of different dimensions of performance, ranging from application to additional task domains, inclusion of additional classes of features within the road following domain, addition of learning capabilities to the system, and so on. This section outlines a number of the possible directions for extension.

While YARF makes a number of contributions to the reliability and competence of visual tracking of the road and its lane structure, the perceptual and path planning capabilities it provides do not address the full needs of the driving task. Further progress towards providing the perceptual support needed for the tactical level of the driving task will require incorporating feature trackers to locate additional types of objects. The introduction discussed the model of tactical driving constructed by Reece, and the set of underlying perceptual routines assumed by his ULYSSES system. Implementing the perceptual routines whose competence is not already provided by YARF would require routines to locate the following classes of objects: markings on the pavement such as turn arrows and crosswalks; various types of road signs, in particular speed limit, stop, and yield signs; traffic lights; and vehicles on the road.

The existing YARF feature trackers look in small rectangular or parallelogram shaped windows in order to make a point measurement of a feature location at a particular lookahead distance. Features such as traffic signs are extended objects which would require larger search windows with more arbitrary shapes. The use of standard colors for objects such as stop and yield signs may permit the use of hue-based trackers built using the same type of pixel classification as the yellow stripe tracker, but will require region extraction and shape analysis to perform reliable detection. In addition, recognizing some types of signs may require limited character recognition capabilities.

The map of the road network used to specify missions for YARF is currently very limited. Only two types of landmarks are represented: intersections and changes in lane structure. A fixed set of control annotations are included. The ability to incorporate additional types of landmarks in the map is important for two reasons. The first is that it permits a richer set of mission specifications. Thus, another direction for extending YARF is the construction of a general annotated map facility designed to support missions specified by a mixture of qualitative and quantitative information about the route and the landmarks to be encountered.

The second reason that allowing a richer set of landmarks is important is that a system can take advantage of distinctive landmarks recognized during one run to reduce the perception and planning cost of later retraverses of the same route. An example of this would be adding the information that a particular intersection was a four-way stop into the map after the system encounters the intersection for the first time. Subsequent runs which encountered the same intersection would not have to expend resources checking for the traffic control signs. Incorporating this type of learning from experience will require the integration of algorithms which can judge the utility of adding a given landmark into the map.

Integration with a map-based strategic driving aid such as the *Travelpilot*$^{TM}$ system would allow YARF to take advantage of the existing map database developed for such systems. It would also demonstrate the potential to produce integrated autonomous systems combining vision based road following techniques with non-autonomous driver's aid technologies already under development. Such an extension would require fast algorithms to detect lane structure for roads branching out at intersections as well as for the road the vehicle is on. Extensions to SHIVA combined with a fast parallel implementation would be one possible solution to this problem.

The current YARF map indicates the specific changes in lane structure which will occur at the transition between stripes along the mission route, explicitly stating that road X will change from two lanes to three lanes. An alternative would be to use feature gap analysis to detect generic classes of changes in road structure without a prior map. Examples would be detecting the addition of turn lanes near intersections or the approach of exit ramps based on shifts in lane markings.

A longer term agenda involves performing the driving task based on measurements in the image plane without any explicit reconstruction of scene geometry. This would reduce the number of system parameters whose values need to be measured. The SHIVA algorithm provides an example of this. The only camera calibration parameter needed by SHIVA is the location of the horizon row, compared to the four parameters which need to be measured for the camera model used in constructing the local map. Section 6.5.2 described a possible implementation of the YARF model of road shape in image coordinates. Pure pursuit steering control can be performed based on road location in the image. A number of algorithms have been developed which use image flow to determine time to collision without computing object range. Such an extension would require that the tactical constraints on vehicle behavior be expressible in terms of variable such as time to collision rather than object range and speed. In addition, such a system would not be able to build a local map incorporating information from multiple frames. This would result in a need for more dense sampling of the features in each frame, and modifications to the gap analysis algorithms for detecting stripe transitions. It would also require different algorithms for locating branching roads at intersections.

# 8 . Bibliography

[1] Aubert, Didier; and Thorpe, Chuck. *Color Image Processing for Navigation: Two Road Trackers.* Technical Report CMU-RI-TR-90-09, Robotics Institute, Carnegie Mellon, April, 1990.

[2] Ballard, Dana Harry; and Brown, Christopher M. *Computer Vision.* Prentice-Hall, 1982.

[3] Beckman, R. J.; and Cook, R. D. Outlier.....s. *Technometrics.* 25(2):119-149, May, 1983.

[4] Binford, T. O. Survey of Model-Based Image Analysis Systems. *Int. J. Robotics Research.* 1 1981.

[5] Blaauw, Gerard J. Driving Experience and Task Demands in Simulator and Instrumented Car: A Validation Study. *Human Factors.* 2473-486, 1982.

[6] Bush, Vannever. *Modern Arms and Free Men.* Simon and Schuster, New York, 1949.

[7] Buxton, James L.; Honey, Stanley K.; Suchowerskyj, Wadym E.; and Tempelhof, Alfred. The Travelpilot: A Second-Generation Automotive Navigation System. *IEEE Transactions on Vehicular Technology.* 40(1):41-44, February, 1991.

[8] Catling, I.; and McQueen, B. Road Transport Informatics in Europe -- A Summary of Current Developments. *Proceedings of the 5th Jerusalem Conference on Information Technology.* October, 1990.

[9] Crisman, Jill D.; and Thorpe, Charles E. Color Vision for Road Following. *Vision and Navigation: The Carnegie Mellon Navlab.* In Charles E. Thorpe, Kluwer Academic Publishers, 1990, Chapter 2.

[10] Crisman, Jill. *Color Vision for the Detection of Unstructured Roads and Intersections.* PhD thesis, Carnegie-Mellon University, 1990.

[11] DeMenthon, Daniel; and Davis, Larry. Reconstruction of a Road by Local Image Matches and Global 3D Optimization. *Proceedings 1990 IEEE International Conference on Robotics and Automation.* May, 1990.

[12] Dickmanns, Ernst Dieter; and Graefe, Volker. Applications of Dynamic Monocular Machine Vision. *Machine Vision and Applications.* 1241-261, 1988.

[13] Dickmanns, Ernst D.; and Mysliwetz, Birger D. Recursive 3-D Road and Relative Ego-State Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 14199-213, 1992.

[14] Dowling, Kevin; Guzikowski, Robert; Ladd, Jim; Pangels, Henning; Singh, Sanjiv; and Whittaker, William. Navlab: An Autonomous Navigation Testbed. *Vision and Navigation: The Carnegie Mellon Navlab.* In Charles E. Thorpe, Kluwer Academic Publishers, 1990, Chapter 12.

[15] Fennema, Claude L. *Interweaving Reason, Action and Perception.* PhD thesis, University of Massachusetts at Amherst, Department of Computer and Information Science, 1991.

[16] Fischler, Martin A.; and Bolles, Robert C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM.* 24(6), June, 1981.

[17] Franke, U.; Fritz, H.; and Mehring, S. Long Distance Driving with the Daimler-Benz Autonomous Vehicle VITA. *PROMETHEUS Workshop.* December, 1991.

[18] Franke, U. Real Time 3D-Road Modeling for Autonomous Vehicle Guidance. *Seventh Scandanavian Conference on Image Analysis.* August, 1991.

[19] Goto, Y.; Matsuzaki, K.; Kweon, I.; and Obatake, T. CMU Sidewalk Navigation System: A Blackboard-Based Outdoor Navigation System Using Sensor Fusion with Colored-Range Images. *Proc. Fall Joint Computer Conference.* November, 1986.

[20] Hashimoto, K.; Nakayama, S.; Saito, T.; Oono, N.; Ishida, S.; Unoura, K.; Ishii, J.; and Okada, Y. An Image-Processing Architecture and a Motion Control Method for an Autonomous Vehicle. *Proceedings of the Intelligent Vehicles '92 Symposium.* June, 1992.

[21] Hattori, A.; Hosaka, A.; and Taniguchi, M. Driving Control System for Autonomous Vehicle Using Multiple Observed Point Information. *Proceedings of the Intelligent Vehicles '92 Symposium.* June, 1992.

[22] Jochem, Todd; Pomerleau, Dean; and Thorpe, Charles. *Proceedings, Intelligent Autonomous Systems 3.* February, 1993.

[23] Juberts, M.; Raviv, D.; and Bishop, J. Richard. Autonomous Road Following: A Vision-Based Approach for AVCS. *Proceedings, Intelligent Autonomous Systems 3.* February, 1993.

[24] Jurgen, Ronald K. Smart Cars and Highways go Global. *IEEE Spectrum.* 26-36, May, 1991.

[25] Kanatani, Kenichi; and Watanabe, Kazunari. Reconstruction of 3-D Road Geometry from Images for Autonomous Land Vehicles. *IEEE Transactions on Robotics and Automation.* 6(1):127-132, February, 1990.

[26] Kender, John R.; and Leff, A. Why Direction-Giving is Hard: The Complexity of Using Landmarks in One-Dimensional Navigation. *IEEE Transactions on Systems, Man and Cybernetics.* 19(6), November/December, 1989.

[27] Kenue, Surender K. LANELOK: Detection of Lane boundaries and Vehicle Tracking Using Image-Processing Techniques -- Part I: Hough-Transform, Region Tracing and Correlation Algorithms. *SPIE Mobile Robots IV.* 1989.

[28] Kenue, Surender K. LANELOK: Detection of Lane boundaries and Vehicle Tracking Using Image-Processing Techniques -- Part II: Template Matching Algorithms. *SPIE Mobile Robots IV.* 1989.

[29] Kenue, Surender K. LANELOK: Detection of Lane boundaries and Vehicle Tracking Using Image-Processing Techniques -- Parts I and II. *SPIE Mobile Robots IV.* 1989.

[30] Kluge, Karl; and Thorpe, Charles E. Explicit Models for Robot Road Following. *Vision and Navigation: The Carnegie Mellon Navlab.* In Charles E. Thorpe, Kluwer Academic Publishers, 1990, Chapter 3.

[31] Kuan, Darwin; Phipps, Gary; and Hsueh, A.-Chuan. Autonomous Land Vehicle Road Following. *Proceedings First International Conference on Computer Vision.* June, 1987.

[32] Kushner, Todd R.; and Puri, S. Progress in Road Intersection Detection for Autonomous Vehicle Navigation. *Mobile Robots II.* November, 1987.

[33] Liou, Shih-Ping; and Jain, Ramesh. Road Following Using Vanishing Points. *Computer Vision, Graphics, and Image Processing.* 39116-130, 1987.

[34] Mettala, Erik G. The OSD Tactical Unmanned Ground Vehicle Program. *Proceedings DARPA Image Understanding Workshop.* 1992.

[35] Michon, J. A. A Critical View of Driver Behavior Models: What Do We Know, What Should We Do?. *Human Behavior and Traffic Safety.* In L. Evans and R. Schwing, Plenum, 1985.

[36] Morgenthaler, David G.; Hennessy, Steven J.; and DeMenthon, Daniel. Range-Video Fusion and Comparison of Inverse Perspective Algorithms in Static Images. *IEEE Transactions on Systems, Man, and Cybernetics.* 20(6):1301-1312, November/December, 1990.

[37] Mysliwetz, Birger D.; and Dickmanns, E. D. Distributed Scene Analysis for Autonomous Road Vehicle Guidance. *Proceedings SPIE Conference on Mobile Robots.* November, 1987.

[38] Ozaki, Tohru; Ohzora, Mayumi; and Kurahashi, Keizou. An Image Processing System for Autonomous Vehicle. *SPIE Mobile Robots IV.* 1989.

[39] Polk, Amy; and Jain, Ramesh. A Parallel Architecture for Curvature-Based Road Scene Classification. *Roundtable Discussion on Vision-Based Vehicle Guidance '90 (in conjunction with IROS)*. July, 1990.

[40] Pomerleau, Dean A. *Neural Network Perception for Mobile Robot Guidance*. PhD thesis, Carnegie Mellon University, 1992.

[41] Ponce, Jean. On Characterizing Ribbons and Finding Skewed Symmetries. *Computer Vision, Graphics, and Image Processing*. 52(3):328-40, December, 1990.

[42] Reece, Douglas A. *Selective Perception for Robot Driving*. PhD thesis, Carnegie Mellon University, 1992.

[43] Rousseeuw, Peter J.; and Leroy, Annick M. *Robust Regression and Outlier Detection*. John Wiley & Sons, Inc., 1987.

[44] Schaaser, L. T.; and Thomas, B. T. Finding Road Lane Boundaries for Vision Guided Vehicle Navigation. *Vision-Based Vehicle Guidance*. In Ichiro Masaki, Springer-Verlag, 1992, Chapter 11.

[45] Shladover, S. E. The California PATH Program of IVHS Research and Its Approach to Vehicle-Highway Automation. *Proceedings of the Intelligent Vehicles '92 Symposium*. June, 1992.

[46] Stengel, R.; and Niehaus, A. Intelligent Guidance for Headway and Lane Control. *Proceedings of the 1989 American Control Conference*. 1989.

[47] Stentz, Anthony. *The NAVLAB System for Mobile Robot Navigation*. PhD thesis, Carnegie Mellon University, 1990.

[48] Suzuki, A.; Yasui, N.; Nakano, N.; and Kaneko, M. Lane Recognition System for Guiding of Autonomous Vehicle. *Proceedings of the Intelligent Vehicles '92 Symposium*. June, 1992.

[49] Thorpe, Charles E. *Vision and Navigation: The Carnegie Mellon Navlab*. Kluwer Academic Publishers, 1990.

[50] Thorpe, Charles; and Gowdy, Jay. Annotated Maps for Autonomous Land Vehicles. *Proceedings of the DARPA Image Understanding Workshop*. 1990.

[51] Thorpe, Charles; Amidi, Omead; Gowdy, Jay; Hebert, Martial; and Pomerleau, Dean. Integrating Position Measurement and Image Understanding for Autonomous Vehicle Navigation. *Proceedings of the Second International Workshop on High Precision Navigation*. 1991.

[52] Thorpe, Charles; Hebert, Martial; Kanade, Takeo; and Shafer, Steven. Towards Autonomous Driving: The CMU Navlab. Part I -- Perception. *IEEE Expert*. 31-42, August, 1991.

[53] Thorpe, Charles; Hebert, Martial; Kanade, Takeo; and Shafer, Steven. Towards Autonomous Driving: The CMU Navlab. Part II -- Architecture and Systems. *IEEE Expert.* 44-52, August, 1991.

[54] Turk, Matthew A.; Morgenthaler, David G.; Gremban, Keith D.; and Marra, Martin. VITS -- A Vision System for Autonomous Land Vehicle Navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 10(3), May, 1988.

[55] U. S. Bureau of the Census. *Statistical Abstract of the United States: 1988.* U. S. Bureau of the Census, 1987.

[56] Wallace, Richard; Stentz, Anthony; Thorpe, Charles; Moravec, Hans; Whittaker, William; and Kanade, Takeo. First Results in Robot Road Following. *Proceedings IJCAI 85.* August, 1985.

[57] Waxman, Alan; LeMoigne, Jacqueline; Davis, Larry; Srinivasan, B.;Kushner, Todd; Liang, E.; and Siddalingaiah, T. A Visual Navigation System for Autonomous Land Vehicles. *Journal of Robotics and Automation, Vol. 3.* 1987.